Oracle® Configurator

Oracle Configurator Custom Web Deployment Guide

Release 11*i*

September 2000

Part No. A77067-06

Oracle Configurator (OC) enables you to integrate the Oracle Configurator into your own Internet application. This document describes how to embed the OC into an Internet host application, such as a custom web store.

ORACLE®

Oracle Configurator Custom Web Deployment Guide, Release 11*i*

Part No. A77067-06

Primary Author: Mark Sawtelle

Contributing Authors: Tina Brand

Contributors: Brent Benson, Mike Colena, David Gosselin, CK Jeyaprakash, Manoj Jose, David Kulik, David Lee, Keri-Lyn Mullis, Janet Page, Marty Plotkin, Mike Sheehy, Sean Tierney, Lois Wortley

Contents

Send Us Your Comments xi

Preface xiii

Intended Audience xiii
Technologies and Tools xiii
Related Documents xiv
Structure xiv
Conventions xiv

1 Introduction

Typical Tasks 1–1
Oracle Configurator Architecture 1–2
Three-tier Web Application Architectures 1–2
Application Flow 1–5
Elements Needed and Provided 1–5

2 Servlet Files

Related Documentation 2–1
Prerequisites 2–1
Installing the Required Files 2–2
Required Files and Locations 2–2
General Directory Structure 2–2
File Types 2–4
Files for the Servlet Directory 2–4

Files for the HTML Directory 2–5
Files for the Media Directory 2–6

3 Integration: Session Initialization

Overview of this Chapter 3–1
How it Works 3–1
What You Do 3–2
Responsibilities of the Host Application 3–3
Definition of Session Initialization 3–3
Setting Parameters 3–4
Parameter Syntax 3–4
Typical Parameter Values 3–5
Minimal Test of Initialization 3–6
Parameter Validation 3–7
Initialization Parameter Types 3–7
Connection Parameters 3–8
Configuration Identification Parameters 3–8
Identifying the User Interface Definition 3–9
Identifying the Configuration 3–9
Identifying the Model 3–10
Return URL Parameter 3–10
Pricing Parameters 3–11
Pricing Parameter Precedence 3–11
Oracle Applications Release 11*i* Pricing Parameters 3–12
Oracle Applications Release 10.7 and 11.0 Pricing Parameters 3–12
ATP Parameters 3–13
ATP Method Precedence 3–13
Oracle Applications Release 11*i* ATP Parameters 3–13
Oracle Applications Release 10.7 and 11.0 ATP Parameters 3–14
Arbitrary Parameters 3–14
Initialization Parameter Descriptions 3–14

4 Integration: Session Termination

How it Works 4–1
What You Do 4–1

Definition of Session Termination 4–1
XML Message Structure 4–2
Submission 4–3
Configuration Status 4–4
Configuration Outputs 4–5
Configuration Messages 4–7
Cancellation 4–8
Error 4–8
The Return URL 4–9
Specifying the Return URL 4–9
Implementing the Return URL 4–10

5 Integration: Batch Validation

How it Works 5–1
Passing the Batch Validation Message 5–1
Calling the CZ_CF_API.VALIDATE Procedure 5–2

6 Integration: Pricing and ATP

How it Works 6–1
What You Do 6–2
Database Compatibility 6–3
Installation Scripts 6–4
Implementation Scripts 6–4
The Pricing Callback Interface 6–5
Use of the Database in the Price Multiple Items Procedure 6–7
Examples of the Pricing Callback Interface 6–8
The ATP Callback Interface 6–9
Use of the Database with the ATP Callback Interface 6–9
Examples of the ATP Callback Interface 6–11
Initialization Parameters 6–11
Testing Pricing and ATP Integration 6–13

7 User Interface

How it Works 7–1

Structure of the HTML Template 7–1
The Header Frame 7–4
The Left and Right Border Frames 7–5
What You Do 7–5
The Default Configuration Window 7–6
Customizing the User Interface in Oracle Configurator Developer 7–6
Restrictions on the Configuration Window 7–6
Customizing the HTML Templates 7–7
Specifying the Location of Media Files 7–7
Hiding the Model Tree 7–8
Modifying the Header Bar 7–8

A The Checkout Servlet

B Examples of Pricing and ATP Callback Procedures

C The Header HTML Template

Glossary of Terms

Glossary of Acronyms

Index

List of Examples

3–1	Syntax of intialization message in HTML context	3–4
3–2	Basic XML initialization parameters	3–5
3–3	Minimal HTML for invoking the configuration window	3–6
4–1	Example of structure of termination message	4–2
4–2	Configuration outputs in the termination message	4–5
4–3	Configuration messages in the termination message	4–7
5–1	Example of batch validation message	5–2
5–2	Example of Calling the CZ_CF. _API.VALIDATE Procedure	5–3
6–1	Pricing Callback Interface	6–8
6–2	ATP Callback Interface	6–11
6–3	Initialization message using 11*i* pricing and ATP parameters	6–12
A–1	The default Checkout servlet (Checkout.java)	A–1
B–1	Example of Single-item Callback Pricing Procedure	B–1
B–2	Example of Multiple-item Callback Pricing Procedure	B–2
B–3	Example of Callback ATP Procedure	B–2
C–1	The Header Frame template file (czHeader.htm)	C–1

List of Figures

1–1 Architectural Overview of Oracle Configurator 1–4
6–1 Pricing Architecture in Oracle Configurator 6–2
6–2 Pricing Data in the Configuration Window 6–13
6–3 Pricing and ATP Data in the Summary Pane 6–14
6–4 Pricing and ATP Data in the Submission Display 6–15
7–1 The Structure of the HTML Template 7–2
7–2 The czHeader frame 7–4

List of Tables

1–1 Three-tier web application architecture 1–2
1–2 Elements and Actions Needed and Provided in Oracle Configurator 1–6
2–1 Overview of tasks for installing the servlet 2–1
2–2 Default Installation Methods for the Oracle Configurator window 2–2
2–3 General Structure of Directories for OC 2–3
2–4 OC Directories for Windows NT 2–3
2–5 File Types 2–4
2–6 Files for the Servlet Directory 2–4
2–7 Files for the HTML Directory 2–5
2–8 Files for the Media Directory 2–6
3–1 Explanation of initialization parameters in Example 3–2 3–5
3–2 Types of Initialization Parameters 3–7
3–3 Configuration Identification Parameters 3–8
4–1 Termination conditions 4–1
5–1 Parameters for the CZ_CF_API.VALIDATE procedure 5–3
6–1 Price Single Item Procedure Parameters 6–5
6–2 Price Multiple Items Procedure Parameters 6–6
6–3 CZ_PRICING_STRUCTURES Database Table 6–7
6–4 ATP Procedure Parameters 6–9
6–5 CZ_ATP_REQUESTS Database Table 6–10
7–1 Constraints on the HTML Template 7–3
7–2 Browser-Specific Frameset Files 7–4
7–3 Buttons in the Header Frame 7–5

Send Us Your Comments

Oracle Configurator Custom Web Deployment Guide, Release 11*i*

Part No. A77067-06

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments through your call to Oracle Support Services or by sending them to:

Oracle Configurator
Oracle Corporation
Documentation
21 North Avenue
Burlington, MA 01803
USA

If you would like a reply, please give your name, address, and telephone number below.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Oracle Configurator (OC) is an Oracle product that enables end users of a web-based application to configure products and services.

This manual describes how to embed the Oracle Configurator window into an Internet application, such as a custom web store.

You can see the Oracle Configurator window in use in a number of Oracle's Internet applications, such as Oracle iStore and Sales Online.

Intended Audience

This manual is intended primarily for developers of custom web stores.

Technologies and Tools

Working with the OC requires varying degrees of familiarity with the following technologies and tools:

- web browsers
- Internet application servers (web servers)
- servlets
- web stores
- Java
- JavaScript
- DHTML
- HTML

- XML and XSL

Related Documents

For more information, see the following manuals in the Oracle Configurator documentation set:

- *Oracle Configurator Developer User's Guide*
- *Oracle Configurator Developer Tutorial*
- *Oracle Configurator and SellingPoint Administration Guide*
- *Oracle Configuration Interface Object (CIO) Developer's Guide*

The following documents may also be useful:

- *Oracle8i JDBC Developer's Guide and Reference*

Structure

This manual contains the following chapters and appendices:

Chapter 1, "Introduction"

Chapter 2, "Servlet Files"

Chapter 3, "Integration: Session Initialization"

Chapter 4, "Integration: Session Termination"

Chapter 6, "Integration: Pricing and ATP"

Chapter 7, "User Interface"

Appendix A, "Troubleshooting Servlet Installation"

Appendix A, "The Checkout Servlet"

Appendix B, "Examples of Pricing and ATP Callback Procedures"

Appendix C, "The Header HTML Template"

Conventions

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
italics	Italic type in text or tables indicates user-supplied text. Replace these placeholders with a specific value or string.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

Introduction

Typical Tasks

This document will help you complete the following tasks, which you must perform in order to integrate the Oracle Configurator (OC) into your internet host application:

	For this task ...	See ...
Develop the Configuration Window	In Oracle Configurator Developer, construct your Model and Rules.	*Oracle Configurator Developer User's Guide*
	In Oracle Configurator Developer, create and modify your User Interface.	
	Test Model, Rules, and User Interface in Oracle Configurator Developer	

	For this task ...	See ...
Customize and Deploy the web-based Configuration Window	Install the Oracle Configurator servlet on your internet application server.	Chapter 2, "Servlet Files"
	Tailor the initialization message that invokes the configuration window and the return URL that handles the configuration outputs.	Chapter 3, "Integration: Session Initialization"
	Extract the required data from the XML termination message produced by the configuration window.	Chapter 4, "Integration: Session Termination"
	Provide pricing and ATP data in the configuration window.	Chapter 6, "Integration: Pricing and ATP"
	Customize certain HTML files to modify the appearance of the configuration window.	Chapter 7, "User Interface"
	Test the configuration window in the host application, running in an internet browser.	Appendix A, "Troubleshooting Servlet Installation" and *Oracle Configurator Developer User's Guide*

Oracle Configurator Architecture

Three-tier Web Application Architectures

Oracle Configurator allows you to deploy a web-based configuration window in a three-tier architecture:

Table 1–1 Three-tier web application architecture

Tier	Tier components	OC Components
Client	thin client	your host application, running in a browser
		OC configuration window (DHTML)
Application	application (web) server	Oracle Apache
	application logic	Oracle Configurator Active Model

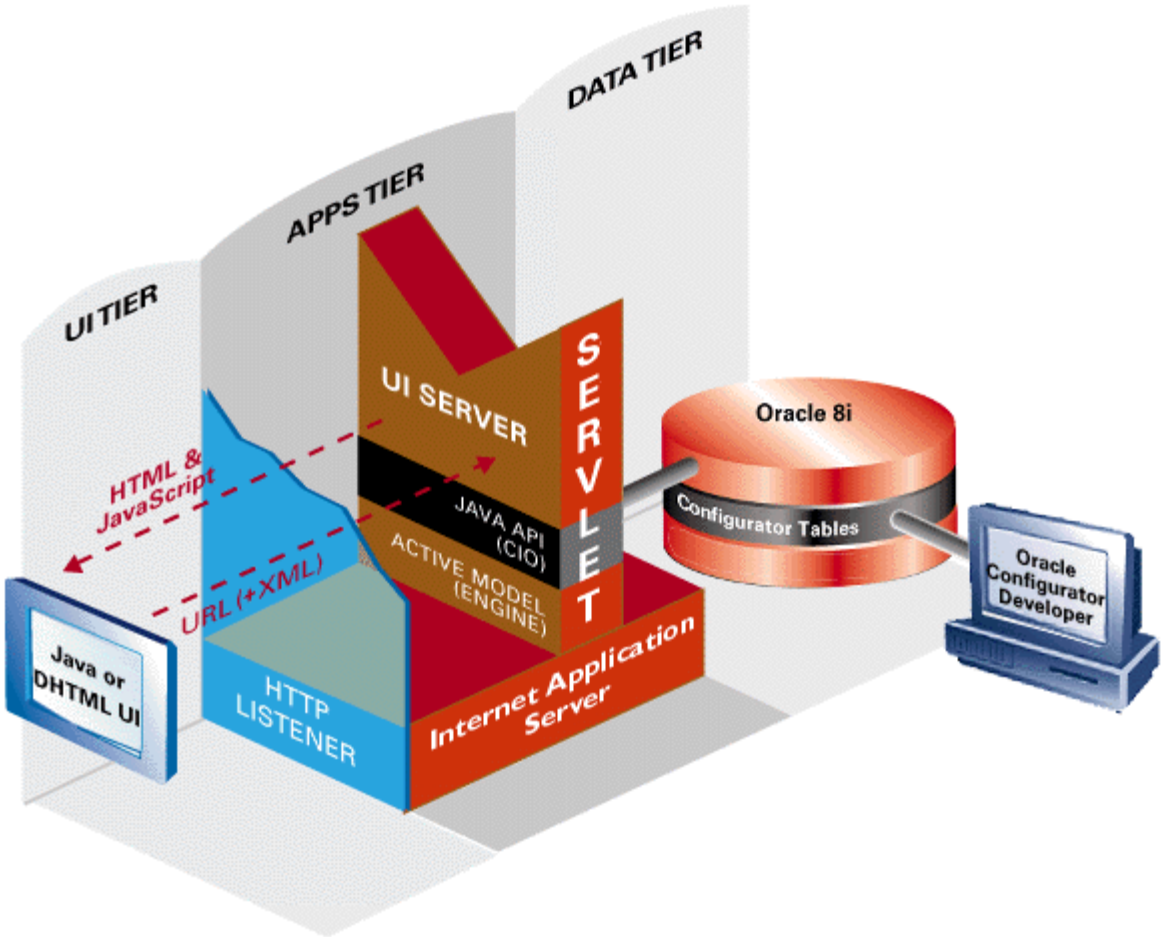
Table 1–1 Three-tier web application architecture

Tier	Tier components	OC Components
Database	application database	Oracle8*i* Enterprise Edition, Oracle Configurator schema

In the client tier, a host application running in a browser displays the configuration window, which in turn communicates with the Oracle Configurator engine and the Oracle Configurator schema.

An application server brokers the HTML, XML, and data queries that provide the configuration window with content from the Oracle Configurator schema. A web server brokers the connection (HTTP).

Figure 1–1 Architectural Overview of Oracle Configurator



The UI Servlet is installed on the application server, along with the other server-side components of Oracle Configurator. The database should not be on the same machine.

The application server is the critical component that enables the thin-client browser to maintain a web interface to the Configuration Model accessed by the configuration window. The servlet running on the application server interfaces with

the UI Server. The UI Server generates a JavaScript rendering of the Active Model and Active UI.

Application Flow

1. An initialization message is sent to the UI Servlet. The initialization includes parameters that specify a database connection (such as user, password, database) and parameters that identify the configuration model required to drive the configuration window (such as an ID for the model, or a particular saved configuration).
2. If a configurator session is not already underway for this user, the servlet creates a new UI session.

   On initial load, all mandatory components are loaded into memory and remain in memory for the entire session. Optional configurable components are loaded on demand and will be unloaded from memory as required.
3. The UI Servlet holds the session information in its session object for that user, and passes all XML messages to the UI Server.
4. The UI Server produces JavaScript.
5. JavaScript is substituted into the HTML template file as the response to the user's request, and displayed in the client browser.
6. Messages from the configuration window (client) to the UI Server are generic input events (click or data input) on a particular control ID within a particular session, stored in the initialization message. Results of a processed event are rendered as JavaScript, to describe changes to the client configuration window such as navigation results, altered logic states, changed counts, etc.

Elements Needed and Provided

Table 1–2 lists all the elements needed by a web application using Oracle Configurator, with a brief description of how you use each in building your host application that incorporates the Oracle Configurator configuration window.

Table 1–2 Elements and Actions Needed and Provided in Oracle Configurator

Element Needed	Element Provided?	Action Required?	Action
Your host application (e.g., Oracle iStore or other web store)	No	Yes	You implement the application.
Web server (e.g., Apache)	No	Yes	You install and maintain the server.
Oracle Applications database (including Oracle Configurator schema tables)	No	Yes	You provide connectivity between the database and the UI Servlet.
Oracle Configurator UI Servlet	Yes	No	The UI Servlet encapsulates other elements: ▪ the UI Server ▪ the CIO ▪ the logic engine
Oracle Configurator UI Server	Yes	No	Automatically accessed by UI Servlet to process user interface
Oracle Configurator CIO	Yes	No	Automatically accessed by UI Servlet. (Also available for custom programming.)
Oracle Configurator logic engine	Yes	No	Automatically accessed by CIO, through UI Servlet.
DHTML configuration window	Yes	No	Configuration window is produced automatically by UI Servlet.
Configuration window UI layout	Yes	No	Layout of UI elements is automatically determined by Model, stored in Oracle Configurator schema.
Default HTML templates for configuration window	Yes	No	Configuration window automatically displays UI in default format.
Customized HTML templates for configuration window	No	Yes	You modify the customizable elements of HTML templates to modify appearance of configuration window, or create new templates following documented structure.

Table 1–2 (Cont.) Elements and Actions Needed and Provided in Oracle Configurator

Element Needed	Element Provided?	Action Required?	Action
Image files, for the background and UI controls of the configuration window.	Yes	No	Image files appear in HTML templates and JavaScript controls. (You can customize the set of files.)
XML initialization message for the UI Servlet	No	Yes	You specify the XML parameters in the initialization message that your application passes to the configuration window.
XML termination message from the UI Servlet	No	Yes	You parse the XML output from the UI Server's termination message, and pass data back to your host application.
"Return URL" Java servlet	No	Yes	You implement a servlet class that provides behavior after the configuration window is used.
Integration code for order entry API	No	Yes	You implement the integration code.

2

Servlet Files

In order to use the Oracle Configurator window, you must first install the UI Servlet on your Internet server. This process consists of the following tasks:

Table 2–1 Overview of tasks for installing the servlet

For This Task ...	See ...
Installing the required files.	Installing the Required Files on page 2-2
Moving files to alternate locations. (Optional)	Required Files and Locations on page 2-2

Related Documentation

You will need to consult the following documentation when installing the Oracle Configurator window servlet:

- *Oracle Configurator Installation Guide*
- *Oracle Configurator ReadMe*
- *Oracle Configurator Release Notes*
- *Oracle Configurator and SellingPoint Administration Guide*

Prerequisites

- You must have installed Oracle Applications Release 11*i*, specifically including Oracle Configurator.
- You must have installed your internet server. Oracle Configurator has been tested with Oracle Internet Application Server (iAS).

- You must have JDBC connectivity between your internet server and your Oracle database.
- Check the *Oracle Configurator Release Notes*, and the chapter on deployment in the *Oracle Configurator and SellingPoint Administration Guide*, for any other requirements.

Installing the Required Files

The required files must be in place before installing the servlet in your Internet server. This is normally done automatically, as described in Table 2–2.

- You may wish to move certain files to other locations, to suit the needs of your site or host application. If so, see Required Files and Locations on page 2-2.

Table 2–2 Default Installation Methods for the Oracle Configurator window

Release	Platform	To install the required files...
Release 11*i*	Windows NT Solaris	On a Windows NT machine, run the setup program on the Oracle Configurator CD (SPSetup.exe).
Release 11*i*	Solaris	On a Solaris machine, run Oracle RapidInstall.

Required Files and Locations

There may be reasons for moving certain installed files to alternate locations.

- To suit the needs or working conventions of your site.
- To suit the needs of your host application.

In order to assist you in moving these files correctly, this section describes constraints and guidelines on their location.

The files are also enumerated, in order to assist you in troubleshooting simple problems arising from missing or misplaced files.

General Directory Structure

Table 2–3 shows the directories required for the Oracle Configurator window, and their relationship. This general structure applies to all platforms, though the details may vary by platform. In some cases, the same physical directory may fill more than one role.

Table 2–3 General Structure of Directories for OC

Directory Role	Description
OC Installation	This directory is referenced in these instructions as *oc_install*. The directory in which you install OC, based on your choice of installation directory in the Oracle Configurator setup program. On Windows NT, this defaults to ORACLE_HOME\OSP. On Solaris, this defaults to $APPL_TOP.
Servlet	Contains the Java class or archive files that implement the UI Servlet.
HTML	Contains the HTML template files that define the Oracle Configurator window of your host application (see Chapter 7).
Media	Contains the image files used by the Oracle Configurator window of your host application.
Log	Contains log files written by the UI Servlet when the Oracle Configurator window is used.

Table 2–4 provides examples of this general directory structure and naming applied to the Windows NT platform. For information on the directory structure on the Solaris platform, see the *Oracle Configurator Installation Guide*.

Note that it is not strictly necessary for the Servlet directory to have a separate physical location, since the files it contains are referenced by environment variables that you set while installing the Oracle Configurator window servlet.

Directory and file names are case-sensitive on Solaris.

Certain HTML files are language-specific, and are stored in language-specific directories, such as OA_HTML/US. See Table 2–7 on page 2-5 for a list of these files.

Table 2–4 OC Directories for Windows NT

Directory	Example
OC Install	D:\orant\OSP\
Servlet	D:\orant\OSP\OSP\
HTML	D:\orant\OSP\WebUI\OA_HTML\
Media	D:\orant\OSP\WebUI\OA_MEDIA\
Log	C:\temp\

File Types

Table 2–5 shows the types of files included in an installation of the Oracle Configurator window.

Table 2–5 File Types

Type	Filename Extension	Description
HTML Template	.htm, .html	HTML files that provide the layout and interface for the configuration window. Certain of these can be customized.
Java Archive	.jar, .zip	Java archive files that implement the UI Servlet, UI Server, configuration engine, and other elements of the Oracle Configurator window.
Java Class	.class	Java class files that implement application logic.
Java Source	.java	Java source code files that can be modified to customize the behavior of the UI Servlet with your host application.
JavaScript Control	.js	JavaScript control files that implement the behavior of the Oracle Configurator window.
Media	.gif, .jpg	Image files used by your host application.
Shared Object	.so, .dll	The platform-specific implementation of the servlet.

Files for the Servlet Directory

Table 2–6 shows the files that should be installed in your Servlet directory.

The Servlet directory contains files that must be referenced in the PATH and CLASSPATH environment variables. It is not strictly necessary for the Servlet directory to have a distinct physical location. For instance, its files can be located directly in the *`oc_install`* directory, rather than in a separate `/servlets` subdirectory.

For the physical location of the Servlet directory, and examples by platform, see Table 2–4 on page 2-3 and the *Oracle Configurator Installation Guide*.

Table 2–6 Files for the Servlet Directory

File	For Platform	Comment
Java Classes		

Table 2–6 (Cont.) Files for the Servlet Directory

File	For Platform	Comment
apps.zip	Solaris	Includes all the Java classes required for OC, in addition to those for Oracle Applications.
xmlparserv2.zip	Solaris	Includes the Java classes for the XML parser.
cz3rdpty.jar	Windows NT	Includes the Java classes for collections, the XML parser, the Swing UI, and various fundamentals.
config.jar	Windows NT	Includes the Java classes in the package oracle.apps.cz.servlet that implement the UI Servlet, notably these: UiServlet, Proxy, and Logout.
`jdbc111.zip`	All	Includes the Java classes for JDBC database connectivity.
Shared Object		
cz.dll	Windows NT	Must be in the PATH system environment variable on the host machine on which the servlet is installed. This should be set by the OC installation program.
czjni.dll	Windows NT	
libcz.so	Solaris	Must be in the LD_LIBRARY_PATH environment variable parameter for your servlet.
libczjni.so	Solaris	

Files for the HTML Directory

The files listed in Table 2–7 must be located in the HTML directory, on all platforms.

For the physical location of the HTML directory, and examples by platform, see Table 2–4 on page 2-3 and the *Oracle Configurator Installation Guide*.

Table 2–7 Files for the HTML Directory

File
HTML Templates
czCntnt.htm
czdisp.htm
czSource.htm
cztree.htm
Language-Specific (NLS) HTML Templates

Table 2–7 (Cont.) Files for the HTML Directory

File
czFraIE.htm
czFraNS.htm
czHeader.htm
czLeft.htm
czRight.htm
JavaScript Controls
czAll.js

If you are implementing Oracle Configurator in a language other than English, you may need to translate the NLS files.

Files for the Media Directory

The files listed in Table 2–8 must be located in the Media directory, on all platforms.

Table 2–8 Files for the Media Directory

czAvail0.gif	czFCUnSa.gif	czSIbox1.gif
czAvail1.gif	czfldcls.gif	czSIdn0.gif
czBack0.gif	czfldclu.gif	czSIdn1.gif
czBack1.gif	czfldopn.gif	czSIlt0.gif
czblank.gif	czflopnu.gif	czSIlt1.gif
czblbar.gif	czFOpen.gif	czSIrt0.gif
czbollf.gif	czForw0.gif	czSIrt1.gif
czbollt.gif	czForw1.gif	czSIup0.gif
czboluf.gif	czFOUnSa.gif	czSIup1.gif
czbolun.gif	czlgblf.gif	czsplogo.gif
czbolut.gif	czlgblt.gif	czSrBg.gif
czButC0.gif	czlgbuf.gif	czSumm0.gif
czButC1.gif	czlgbun.gif	czSumm1.gif
czButL0.gif	czlgbut.gif	czTabBL.gif

Table 2–8 (Cont.) Files for the Media Directory

czButL1.gif	czLNode.gif	czTabBR.gif
czButR0.gif	czMLNode.gif	czTabC.gif
czButR1.gif	czMNode.gif	czTabL.gif
czCanc0.gif	czMore.gif	czTabR.gif
czCanc1.gif	czoptlf.gif	czTNode.gif
czCboArr.gif	czoptlt.gif	czUnsat.gif
czcomlf.gif	czoptuf.gif	czWarn.gif
czcomlt.gif	czoptun.gif	czyellgr.gif
czcomuf.gif	czoptut.gif	
czcomun.gif	czPLNode.gif	
czcomut.gif	czPNode.gif	
czConf0.gif	czSHbar0.gif	
czConf1.gif	czSHbar1.gif	
czDone0.gif	czSHbox0.gif	
czDone1.gif	czSHbox1.gif	
czFClose.gif	czSIbox0.gif	

3

Integration: Session Initialization

Overview of this Chapter

- How it Works on page 3-1
- What You Do on page 3-2
- Definition of Session Initialization on page 3-3
- Setting Parameters on page 3-4
- Initialization Parameter Types on page 3-7
  - Connection Parameters on page 3-8
  - Configuration Identification Parameters on page 3-8
  - Return URL Parameter on page 3-10
  - Pricing Parameters on page 3-11
  - ATP Parameters on page 3-13
  - Arbitrary Parameters on page 3-14
- Initialization Parameter Descriptions on page 3-14

How it Works

In order to prepare a Model so that it can be configured in the configuration window, you must first either:

- Import BOM and item data into the Oracle Configurator schema so that it is available to Oracle Configurator Developer. Importing is described in the *Oracle Configurator and SellingPoint Administration Guide*.

- Use Oracle Configurator Developer to develop a Model and its associated Configuration Rules and a User Interface.

In order to integrate the configuration window into your host application, the application must:

- Use frames, one of which contains the configuration window.
- Provide a means of posting the initialization message.
- Provide a return URL servlet to process results of your user's selections in the configuration window.

In a typical host application (such as a web store), a button, tab, or similar control is coded so that it causes the configuration window to appear, and to contain the appropriate user interface. For the purposes of this explanation, think of this control as "the Configure button". You intend for your user to click it at a point where configuration of a product is required.

Oracle Configurator provides you with:

- A **servlet**, the UI Servlet, that handles interaction between the host application, the configuration window, and the product Model that you define in Configurator Developer (which contains product structure, logic rules, and user interface definitions).
- The host application invokes the URL of the UI Servlet with a query string that contains an XML *initialization message*. It does this by setting the *location* property of the frame where the Configurator is to be displayed. You tailor this initialization message to specify a number of important parameters that govern the behavior of the configuration window. These are described in this chapter.
- A set of **HTML Template files** that interact with the UI Servlet. You can customize these files to suit them to the needs of your host application, or leave them as they are. See Chapter 7, "User Interface" for details.

See Oracle Configurator Architecture on page 1-2 in Chapter 1, "Introduction" for an explanation of the interaction between all these elements.

What You Do

Integrating the configuration window with your host application consists primarily of causing your host application (e.g., through the coding of the "Configure" button) to post the XML initialization message to the UI Servlet.

You must first install the OC UI Servlet, as described in the *Oracle Configurator Installation Guide*, with additional information in Chapter 2, "Servlet Files".

There are two basic situations that the host application must handle when integrating the configuration window:

You need to handle this...	As described in...
Initialization of the configuration window, to prepare it for your user to perform configuration selections.	Definition of Session Initialization on page 3-3
Termination of the configuration window, to return control and results to the host application when your user closes the window.	Definition of Session Termination on page 4-1

Responsibilities of the Host Application

The responsibilities of the host application while the configuration window is in use are:

- Block the host application while the configuration window is in use, disallowing all user input and navigation to or from the configuration window.
- Disable visible functions in the surrounding host application that would confuse the user while interacting with the configuration window.
- Handle the output from the *return URL*, and close the configurator window by resetting its frame's *location* property.

Definition of Session Initialization

Session initialization takes place when your host application invokes the UI Servlet which opens the configuration window.

When you set the parameters of the initialization message in your host application, your parameters handle the types of responsibilities listed in Initialization Parameter Types on page 3-7.

When your host application invokes the configuration window, the initialization message is sent to the UI Servlet, using the HTTP POST method. (POST is used in preference to GET to accommodate the length of the message.)

The initialization message is written in XML, and has `<initialize>` as its document element. You must specify the parameters for `<initialize>` in order to determine the state in which the configuration window will open.

Setting Parameters

You specify `<initialize>` and its parameters as the value of an XML message that is passed to the UI Servlet. The UI Servlet is invoked through its URL, which is determined when you install it (as described in the *Oracle Configurator Installation Guide*). By default, the URL is:

hostname:*port/virtual_path_of_servlet_class*/oracle.apps.cz.servlet.UiServlet

Parameter Syntax

All parameters to the XML initialization message are specified as name-value pairs, using attributes of the `<param>` document element, in the form:

```
<param name="parameter_name">parameter_value</param>
```

Example 3–1 shows the basic syntax for specifying the UI Servlet's URL and the initialization message as you would typically use them in your host application. The parts that you need to modify are typographically emphasized.

Example 3–1 Syntax of intialization message in HTML context

```
...
<form action="servlet_URL" method="post" >
<input type=hidden name="XMLmsg" value=
'<initialize>
<param name="parameter_1_name">parameter_1_value</param>
<param name="parameter_n_name">parameter_n_value</param>
</initialize>'>
<input type="submit" value="Configure">
</form>
...
```

When your user clicks the "Configure" button produced by the second `INPUT` element in Example 3–1, the initialization message is posted to the UI Servlet.

See Example 3–2 for some typical values for the parameters, and Example 3–3 for a test page that puts the values in context.

Be aware that XML permits you to use either single or double quotation marks around the value of an element's attribute, so you might also write:

```
"<initialize>
    <param name='parameter_name'>parameter_value</param>
</initialize>"
```

Typical Parameter Values

Example 3–2 shows an example of a basic set of initialization parameters that cover all of the types of responsibility shown in Table 3–2 on page 3-7.

See Initialization Parameter Descriptions on page 3-14 for the complete list of valid parameters to the initialization message.

See Example 3–1 for the syntax of the initialization message, and Example 3–3 for a test page that puts the values in context.

Example 3–2 Basic XML initialization parameters

```
<initialize>
  <param name="two_task">vis11</param>
  <param name="gwyuid">applsyspub/pub</param>
  <param name="fndnam">apps</param>
  <param name="user">mfg</param>
  <param name="pwd">welcome</param>
  <param name="ui_type">DHTML</param>
  <param name="ui_def_id">3120</param>
  <param
name="return_url">http://www.mysite.com:10130/configurator/Checkout</param>
</initialize>
```

Table 3–1 Explanation of initialization parameters in Example 3–2

Parameter type	Name	Description
Login	two_task	The name of the database instance to connect to. This value can be read from the environment variable TWO_TASK.
Login	gwyuid	The gateway user ID required for authentication with the Oracle Applications protocol.
Login	fndnam	Used by Oracle Configurator to authenticate standard Oracle Applications users through the FND login algorithms.

Table 3–1 (Cont.) Explanation of initialization parameters in Example 3–2

Parameter type	Name	Description
Login	user	The user ID of the login user.
Login	pwd	The password of the login user.
Login	ui_type	The type of web pages to be returned by the UI Servlet.
Configuration Identification	ui_def_id	The UI Definition ID of the User Interface that you created in Oracle Configurator Developer. There are several ways to specify a Configuration choice. See Configuration Identification Parameters on page 3-8.
Return	return_url	The URL of the Java servlet that processes the configurator's termination message.

Minimal Test of Initialization

Example 3–3 shows the HTML for a minimal web page that invokes the configuration window. You can use this test page as a stand-in for your host application.

This example includes the invocation of the UI Servlet as shown in Example 3–1 and the initialization message parameters as shown in Example 3–2.

Example 3–3 Minimal HTML for invoking the configuration window

```
<html>
<head>
<title>Minimal Configurator Test</title>
</head>
<body>
<form
action="http://www.mysite.com:10130/configurator/oracle.apps.cz.servlet.UiServle
t" method="post">
<input type="hidden" name="XMLmsg" value=
'<initialize>
<param name="two_task">vis</param>
<param name="gwyuid">applsyspub/pub</param>
<param name="fndnam">apps</param>
<param name="user">mfg</param>
<param name="pwd">welcome</param>
<param name="ui_type">DHTML</param>
<param name="ui_def_id">3120</param>
<param
name="return_url">http://www.mysite.com:10130/configurator/Checkout</param>
```

```
</initialize>'>
<p>Click button to configure model...
<input type="submit" value="Configure">
</form>
</body>
</html>
```

Parameter Validation

When your host application invokes the UI Servlet, the UI Server validates the parameters of the initialization message.

- There must be a way of connecting to the database, such as the parameters `two_task` or `alt_database_name`.
- There must be a way to choose a Model to be configured, so there must be one of the combinations described in Configuration Identification Parameters on page 3-8.

If there is a problem processing the initialization message, then the UI Server returns a termination object to the UI Servlet, which returns it to the host application or displays the results to your user, through the URL specified in the `return_url` parameter.

Initialization Parameter Types

This section describes the use of the types of initialization parameters listed in Table 3–2 on page 3-7. All of the initialization parameters are described alphabetically in Initialization Parameter Descriptions on page 3-14.

Table 3–2 Types of Initialization Parameters

Type	Required?	Description	See
Login	Yes	Information required for access to the proper data, such as database, user, and password.	Connection Parameters on page 3-8
Configuration	Yes	Identification of the model to be configured, or of the existing configuration to be modified. See Configuration Identification Parameters on page 3-8.	Configuration Identification Parameters on page 3-8

Table 3–2 (Cont.) Types of Initialization Parameters

Type	Required?	Description	See
Return	No, but recommended	Identification of the return URL that handles the results from the configuration window, such as configuration outputs.	Return URL Parameter on page 3-10
Pricing and ATP	No	Identification of the procedures and interfaces to be used for obtaining prices and ATP dates.	Pricing Parameters on page 3-11 ATP Parameters on page 3-13
Other	No	Miscellaneous desired information.	Arbitrary Parameters on page 3-14

Connection Parameters

In order to connect the configuration window to the database, you must specify one of the following parameters or combinations of parameters in your initialization message.

- database_id
- database_id and icx_session_ticket
- alt_database_name, user, and pwd
- two_task, user, pwd, gwyuid, and fndnam

For descriptions of the individual parameters, see Initialization Parameter Descriptions on page 3-14.

Configuration Identification Parameters

There are several different ways in which you can identify the model to be configured, or the existing configuration to be modified. You must use one of the following parameters or combinations of parameters in your initialization message:

Table 3–3 Configuration Identification Parameters

Method for Configuration Identification	Initialization Parameters
Identifying the User Interface Definition	ui_def_id

Table 3–3 (Cont.) Configuration Identification Parameters

Method for Configuration Identification	Initialization Parameters
Identifying the Configuration	config_header_id
	config_rev_nbr
Identifying the Model	context_org_id
	model_id
	config_creation_date

For descriptions of the individual parameters, see Initialization Parameter Descriptions on page 3-14.

Identifying the User Interface Definition

Parameter to specify:

- ui_def_id

Using this parameter will result in creating a new configuration. It is most useful for identifying a Model created entirely in Oracle Configurator Developer. It is also useful for specifying a particular UI out of several that may be available for a Model, whether or not the Model was created entirely in Configurator Developer.

This ID identifies a User Interface created in Configurator Developer. The User Interface includes identification of the Model to be configured (which is associated with Configuration Rules and the Project in Configurator Developer in which they were developed).

Identifying the Configuration

Parameters to specify:

- config_header_id
- config_rev_nbr

Using this combination of parameters will result in restoring an existing configuration.

The configuration header ID is the main identifier of an existing configuration record previously created and saved by your host application or another application that knows how to save configurations to the Oracle Configurator schema, such as the Oracle SellingPoint application. The configuration revision number

distinguishes among particular saved configurations sharing the same header information.

Identifying the Model

Parameters to specify:

- context_org_id
- model_id
- config_creation_date

Using this combination of parameters will result in creating a new configuration. It is only useful for identifying a Model that was originally created in another application (such as Oracle Applications Bills of Materials) and then imported into Oracle Configurator Developer.

Your host application must determine which Model to configure and be able to identify it by inventory_item_id and organization_id.

Return URL Parameter

The return URL is the fully qualified URL of a Java servlet installed on your web server that implements the behavior that you want after the user has ended the configuration session.

- return_url

```
<param
name="return_url">http://www.mysite.com:10130/configurator/Checkout</param>
```

The example parameter above comes from Example 3–3, "Minimal HTML for invoking the configuration window" on page 3-6.

The URL specification in the `return_url` parameter must stop at the name of the servlet class. You cannot pass parameters to the class in this URL (for instance, with the *`classname`*`?`*`parameter`*`=`*`value`* syntax). The return URL servlet should only get data from the termination message, which is passed to the it as the value of the XMLmsg argument.

The initialization message is sent to the return URL. This occurs in the event of normal termination, cancellation by the end user, or exceptions.

The return servlet is installed in your web server's Servlet directory, as described in Files for the Servlet Directory on page 2-4 in Chapter 2, "Servlet Files".

See The Return URL on page 4-9 for details on the implementation of the return servlet.

Pricing Parameters

See Chapter 6, "Integration: Pricing and ATP" for details on the use of these parameters. See Appendix B, "Examples of Pricing and ATP Callback Procedures" on page B-1 for examples.

Choose from two distinct sets of pricing parameters, depending on which pricing methods are required for your host application:

Pricing method	Use these parameters...
Advanced Pricing (QP), or your own callback pricing procedures that call it	pricing_package_name and the associated parameters listed under Oracle Applications Release 11i Pricing Parameters on page 3-12
Oracle Applications pricing (using a price list ID)	price_list_id and the associated parameters listed under Oracle Applications Release 10.7 and 11.0 Pricing Parameters on page 3-12

If the Oracle Configurator is running in one database (e.g., Release 11*i*), and connecting to another database (e.g., Release 10.7/11.0) to perform pricing, you need to provide the apps_connection_info parameter.

For descriptions of the individual parameters, see Initialization Parameter Descriptions on page 3-14.

Pricing Parameter Precedence

Your initialization message can contain a mix of both types of parameters, but only one set will be used.

- If pricing_package_name is provided in your initialization message, then price_list_id is ignored, even if provided.
- price_list_id is required in order to use Release 10.7 and 11.0 pricing.
- If neither pricing_package_name nor price_list_id is provided, an error is raised.

- When providing pricing_package_name, price_mult_items_proc takes precedence over price_single_item_proc, if both are provided.

Oracle Applications Release 11*i* Pricing Parameters

Since these parameters are designed to be used with an interface using callback procedures, they are also referred to as "callback pricing" parameters.

To use callback pricing, provide these parameters:

- pricing_package_name
- configurator_session_key
- either price_mult_items_proc or price_single_item_proc

See Pricing Parameter Precedence on page 3-11.

Oracle Applications Release 10.7 and 11.0 Pricing Parameters

To use Oracle Applications pricing, provide these parameters:

- price_list_id
- and one or more of the following parameters:
  - agreement_id
  - agreement_type_code
  - customer_id
  - gsa
  - invoice_to_site_use_id
  - order_type_id
  - po_number
  - pricing_flex_field parameters
  - ship_to_site_use_id

See Pricing Parameter Precedence on page 3-11.

These parameters are used when the configuration window calls existing APIs to get pricing and ATP data for configured items.

These parameters are accessible with the Configuration Interface Object (CIO) method `Configuration.setInitParameters()`, described in the API reference portion of the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

ATP Parameters

See Chapter 6, "Integration: Pricing and ATP" for details on the use of these parameters. See Appendix B, "Examples of Pricing and ATP Callback Procedures" on page B-1 for examples.

Choose from two distinct sets of ATP (Available To Promise) parameters, depending on which ATP methods are required for your host application:

If the Oracle Configurator is running in one database (e.g., Release 11*i*), and connecting to another database (e.g., Release 10.7/11.0) to perform ATP calculations, you need to provide the apps_connection_info parameter.

For descriptions of the individual parameters, see Initialization Parameter Descriptions on page 3-14.

ATP Method Precedence

Your initialization message can contain a mix of both types of parameters, but only one set will be used.

If the callback parameters are provided in your initialization message, then they take precedence, and the 10.7/11.0 parameters are ignored, even if provided.

When your end user clicks the "ATP" button, if neither the callback parameters nor the 10.7/11.0 parameters are provided, an error is raised.

Oracle Applications Release 11*i* ATP Parameters

Since these parameters are designed to be used with an interface using callback procedures, they are also referred to as “callback ATP” parameters.

To use callback ATP, provide these parameters:

- atp_package_name
- configurator_session_key
- get_atp_dates_proc
- requested_date (optional, defaults to SYSDATE)
- warehouse_id
- and one of the following:
  - customer_id and customer_site_id
  - ship_to_org_id

Oracle Applications Release 10.7 and 11.0 ATP Parameters

To use existing Oracle Applications ATP procedures, provide all of these parameters:

- application_id
- atp_timeout
- responsibility_id
- user_id

Arbitrary Parameters

You can use the `<param>` document element to send arbitrary parameters that are not already provided, or that may be required for particular applications. You would specify the arbitrary parameter as a name-value pair, using the syntax described in Parameter Syntax on page 3-4:

```
<param name="parameter_name">parameter_value</param>
```

For example:

```
<param name="org_home_page">http://www.oracle.com</param>
```

Such arbitrary parameters are not processed by the UI Server, but are passed to the Oracle Configuration Interface Object (CIO), thus making them available to Functional Companions. (See the *Oracle Configuration Interface Object (CIO) Developer's Guide* for information about obtaining a list of the initialization parameters passed.)

While the architecture of Oracle Configurator allows for the possibility of validating XML parameters against a DTD, this is not currently enforced.

Initialization Parameter Descriptions

This section lists alphabetically all the parameters of the initialization message, which is described in Setting Parameters on page 3-4.

agreement_id
For use with Oracle Applications 10.7 and 11.0 Pricing.

agreement_type_code
For use with Oracle Applications 10.7 and 11.0 Pricing.

alt_database_name
A fully specified JDBC connect string or URL, specifying the JDBC driver and the database alias of the database to connect to. Recommended for use during development of your application, as an alternative to connecting as an Oracle Applications user. Not recommended for production deployment. Must specify thin drivers, for example: `jdbc:oracle:thin:@server01:1521:vis11`.

application_id
The ID from FND_APPLICATION.APPLICATION_ID that is the id of the host application

apps_connection_info
If the Oracle Configurator is running in one database (e.g., Release 11*i*), and connecting to another database (e.g., Release 10.7/11.0) to perform pricing, this parameter describes how to connect to the other database. The apps_connection_info element can contain one of the following parameters or sets of parameters:

- `database_id`
- `database_id` and `icx_session_ticket`
- `user`, `pwd`, `gwyuid`, `fndnam`, and `two_task`
- `alt_database_name`, `user`, and `pwd`

atp_package_name
The name of the PL/SQL interface package that the UI Servlet will call to get ATP information. This parameter is required, if the ATP callback interface is to be used. The particular procedure in the package to be used for calculating ATP dates is specified by `get_atp_dates_proc`.

atp_timeout
The maximum number of seconds that the Oracle Applications 10.7/11.0 ATP calculation method should wait for a result from the server.

calling_application_id
The ID from FND_APPLICATION.APPLICATION_ID that is the id of the host application

config_creation_date
The host application's notion of when the configuration is created.

The value for the `config_creation_date` parameter must be determined by your host application. It is the host application's notion of when the configuration was

created. This date must be in the format "MM-DD- YYYY" where YYYY is a four digit year, MM is a two digit month and DD is a two digit day of the month.

config_header_id
The identifier for an existing configuration. Only used for retrieving a configuration previously saved by the Oracle SellingPoint application. Not present if the configuration was not saved.

The value for the `config_header_id` parameter is obtained from CZ_CONFIG_HDRS.CONFIG_HDR_ID in the Oracle Configurator schema.

config_rev_nbr
The configuration revision number. Only used for retrieving a configuration previously saved by the Oracle SellingPoint application. Not present if the configuration was not saved.

The value for the `config_rev_nbr` parameter is obtained from CZ_CONFIG_HDRS.CONFIG_REV_NBR in the Oracle Configurator schema.

configurator_session_key
An application-dependent string that identifies a configuration session, and allows linking a pricing or ATP request from the configuration window to the host application entity that started the configuration session.Examples for creating this key might be: order header ID with order line ID, or quote ID with quote revision number.

context_org_id
The organization identifier for the BOM exploder. The value for the `context_org_id` parameter must be determined by your host application. It is ultimately derived from MTL_SYSTEM_ITEMS.ORGANIZATION_ID.

customer_id
For use with Oracle Applications 10.7 and 11.0 Pricing. The customer ID from the header of the host application.

customer_id
When getting ATP dates, the ID of the customer to which the configured product is to be shipped.

customer_site_id
When getting ATP dates, the ID of the customer site to which the configured product is to be shipped.

database_id
The name of a DBC file that contains database connectivity information. This file can be found in a standard Oracle Applications installation by calling the PL/SQL function `fnd_web_config.database_id`. If only the database_id is called, it will use the entries batch_validate_user and batch_validate_password to find an Oracle Applications username and password to use for the database connection.

fndnam
Used to establish an Oracle Applications context. Its value can be read from the Forms environment variable FNDNAM. Oracle Configurator authenticates standard Oracle Applications users by using the FND login algorithms. Used in conjunction with `gwyuid`.

get_atp_dates_proc
The name of the "Get ATP Dates" procedure to be called from the package specified by `atp_package_name`. This parameter is conditionally required; it must be provided if the ATP callback interface is to be used.

gsa
For use with Oracle Applications 10.7 and 11.0 Pricing.

gwyuid
Used to establish an Oracle Applications context. Its value can be read from the Forms environment variable GWYUID. Used in conjunction with `fndnam`.

icx_session_ticket
An icx_session_ticket encodes an Oracle Applications session. You should use the PL/SQL function `cz_cf_api.icx_session_ticket` to obtain a value for this parameter. This is the recommended way for Oracle Applications to call the configuration window. When passing an icx_session_ticket, the host application must also pass a `database_id`.

invoice_to_site_use_id
For use with Oracle Applications 10.7 and 11.0 Pricing.

model_id
The inventory item identifier for the top-level model.

The value for the `model_id` parameter must be determined by your host application. It is ultimately derived from MTL_SYSTEM_ITEMS.INVENTORY_ITEM_ID.

model_quantity
The value of this parameter is a number that indicates how many of the model are being configured. If not specified, it defaults to 1. The model quantity may change during a configuration session, so the final quantity should be read from the associated output item in the termination message.

order_type_id
For use with Oracle Applications 10.7 and 11.0 Pricing.

po_number
For use with Oracle Applications 10.7 and 11.0 Pricing.

price_list_id
The identifier for the price list. If not present, pricing information using the Oracle Applications 10.7/11.0 calls will not be available.

price_mult_items_proc
The name of the "Price Multiple Items" procedure to be called from the package specified by `pricing_package_name`. This parameter is conditionally required; either this parameter or `price_single_item_proc` must be provided if pricing callbacks are to be used. This procedure takes precedence over `price_single_item_proc`.

price_single_item_proc
The name of the "Price Single Item" procedure to be called from the package specified by `pricing_package_name`. This parameter is conditionally required; either this parameter or `price_mult_items_proc` must be provided if pricing callbacks are to be used. This procedure will not be called if `price_mult_items_proc` is provided.

pricing_*flex_field*
For use with Oracle Applications 10.7 and 11.0 Pricing. The pricing flexfield values for the model to be configured. These parameters are named `pricing_attribute1` through `pricing_attribute15`.

pricing_package_name
The name of the PL/SQL interface package that the UI Servlet will call to get pricing information. This parameter is required if the pricing callback interface is to be used. The particular procedure in the package to be used for performing pricing is specified by either `price_mult_items_proc` or `price_single_item_proc`.

pwd
The password to use when logging in. Use the Oracle Applications password if you identified the database with the `two_task` parameter. Use the database password if you identified the database with the `alt_database_name` parameter.

read_only
If the value is "true", the UI Server provides a read-only UI for viewing configurations.

requested_date
When getting ATP dates, the requested date entered on the order line. The format of the date must be "MM-dd-yyyy. The default value of SYSDATE is used if you do not specify a different date.

responsibility_id
The ID from FND_RESPONSIBILITY.RESPONSIBILITY_ID.

return_url
The fully qualified URL of a Java servlet installed on your web server that implements the behavior that you want after the user has closed the configuration window. See Return URL Parameter on page 3-10 for details.

save_config_behavior
Value is one of:

`never`	A new configuration will not be saved.
`new_config`	A new configuration will be saved.
`new_revision`	A new revision of the configuration will be saved. (If no existing revision is found, a new configuration will be saved.)
`overwrite`	The existing configuration header and revision will be used.

If the value is `overwrite`, an error will be signalled.

save_usage_behavior
Value is one of:

`never`	A usage will not be saved.
`new_usage`	A new usage will be saved if there is none, and an existing usage will be updated if found.

ship_to_org_id
When getting ATP dates, the ID of the organization to which the configured product is to be shipped. This value is obtained from SHIP_TO_ORG_ID in the OE_ORDER_LINES_ALL table.

ship_to_site_use_id
For use with Oracle Applications 10.7 and 11.0 Pricing.

template_url
A URL to the template file that the configuration window will use when displaying its initial state. It is the responsibility of the host application to choose the correct template, if there need to be multiple templates for multiple languages or browsers. The web page pointed to by the template URL must contain the content frame and the proxy frame. You may need to account for language-specific installation directory names, such as `OA_HTML/US`, when specifying this parameter.

Note: Oracle Configurator provides National Language Support (NLS) (one language at a time), but not Multiple Language Support (MLS) (multiple languages concurrently).

---

terminate_msg_behavior
Value is one of:

`full`	The entire termination message is passed back to the host application. This includes prices, if you have used a pricing interface package (see Chapter 6).
`brief`	No output or messages are passed to the caller.

two_task
The name of the database instance to connect to. This value can be read from the environment variable TWO_TASK.

ui_def_id
The identifier for the User Interface created in Configurator Developer. The value for the `ui_def_id` parameter is obtained from CZ_UI_DEFS.UI_DEF_ID in the Oracle Configurator schema. The value can also be obtained by calling the PL/SQL function `cz_cf_api.ui_for_item`.

ui_type
Type of client. Must be set to `DHTML` when using the Oracle Configurator configuration window in a custom web deployment.

user
The username to use when logging in. Use the Oracle Applications username if you identified the database with the `two_task` parameter. Use the database username if you identified the database with the `alt_database_name` parameter.

user_id
The ID from FND_USER.USER_ID.

warehouse_id
When getting ATP dates, the ID of the organization that is going to ship the configured product to the customer. This value is obtained from SHIP_FROM_ORG_ID in the OE_ORDER_LINES_ALL table.

4

Integration: Session Termination

How it Works

See How it Works on page 3-1, in Chapter 3, "Integration: Session Initialization".

What You Do

See What You Do on page 3-2, in Chapter 3.

Definition of Session Termination

Session termination takes place when the configuration window is closed by one of these conditions:

Table 4–1 Termination conditions

Condition	Example	Explanation
Submission	Your user clicks the Done button.	See Submission on page 4-3
Cancellation	Your user clicks the Cancel button.	See Cancellation on page 4-8
Error	A connection cannot be made to the database.	See Error on page 4-8

When the configuration window is closed, terminating your user's configuration session, the UI Servlet returns the results to your host application in the form of a termination message, written in XML. You need to understand the structure of the termination message in order to be able to extract the desired data from it in your return URL servlet. The structure of this message is described in XML Message Structure on page 4-2.

XML Message Structure

All outputs in the XML termination message are written as XML elements and subelements of the `<terminate>` document element, in the general form:

```
<terminate>

  <element_name>element_value</element_name>

  <element_name>
    <subelement_name>subelement_value</subelement_name>
  </element_name>

</terminate>
```

The top-level structure of the `<terminate>` element is illustrated by these excerpts from its DTD:

```
...
<!ELEMENT terminate (config_header_id?, config_rev_nbr?, valid_configuration?,
complete_configuration?, exit, config_outputs?, config_messages?)>
...
<!ELEMENT config_outputs (output_option*)>
...
<!ELEMENT config_messages (message*)>
...
```

Example 4–1 shows the basic structure of a sample XML termination message. Typographical emphasis and comments have been added to point out the structure; such comments do not appear in actual termination messages.

Example 4–1 Example of structure of termination message

```
<terminate>
  <!-- configuration status elements -->
  <config_header_id>1780</config_header_id>
  <config_rev_nbr>2</config_rev_nbr>
  <valid_configuration>true</valid_configuration>
  <complete_configuration>true</complete_configuration>
  <exit>save</exit>
  <config_outputs>
    <option>
      <component_code>143-1490</component_code>
      <quantity>1</quantity>
      <list_price>0.00</list_price>
      <!-- more elements go here -->
```

```
    </option>
    <!-- more options go here -->
  </config_outputs>
  <config_messages>
    <message>
      <message_type>error</message_type>
      <message_text>Config header does not exist in database.</message_text>
    </message>
    <!-- more messages go here -->
  </config_messages>
</terminate>
```

Submission

Submission occurs after your user closes the configuration window by clicking the "Done" button.

The meaning of the Done button is defined by the context of your host application. For instance, in a web store, it might mean adding the configured product to your user's "shopping cart", or submitting the configured order to your order entry system.

The Done button can be customized, as described in The Header Frame on page 7-4, in Chapter 7, "User Interface".

When the Done button is clicked, the UI Servlet determines whether a return URL has been specified. If so, the servlet it identifies is executed, and the results it generates are passed to your host application for further processing. This is the most important job of the return URL servlet; it captures the configuration selections of your user so that your host application can make use of them. For more details, see The Return URL on page 4-9

After the configuration window is closed, your host application must repaint the frame used by the configuration window.

After submission, the termination message provides the host application with data describing:

- Configuration Status
- Configuration Outputs
- Configuration Messages

Configuration Status

The current configuration status is described by the subelements of `<terminate>` listed in this section.

Subelements

config_header_id
The main identifier of an existing configuration. See the description for config_header_id on page 3-16. This value is displayed in the configuration window with the default label "Configuration Header ID".

config_rev_nbr
The revision number of an existing configuration. See the description for config_rev_nbr on page 3-16. This value is displayed in the configuration window with the default label "Configuration Revision".

valid_configuration
The value will be "true" if no error messages are reported for the configuration. This value is displayed in the configuration window with the default label "Configuration Valid".

complete_configuration
The value will be "true" if all mandatory option classes (required features) are satisfied. This value is displayed in the configuration window with the default label "Configuration Complete".

exit
Value is one of:

`save`	If the configuration was saved.
`cancel`	If the configuration was cancelled.
`error`	If an error was detected while executing in the UI Server.
`processed`	If a batch validation message was processed but not saved.

This value is displayed in the configuration window with the default label "Exit Status".

prices_calculated_flag

Prices are calculated when the user clicks the "Summary" button. This element tells the host application whether this calculation has happened in synchronization with the configuration. If the value of this element is:

`true` The configuration has not been changed since the end user clicked the "Summary" button. That is, the calculated prices are still in synchronization with the configuration.

`false` Prices were not calculated after the configuration had been changed.

This could happen if the end user had never clicked the "Summary" button before clicking "Done", or if the user changed the configuration and did not click the "Summary" button before clicking "Done".

In this case, the host application should reprice each configuration line, to ensure that the proper prices are applied to the configuration.

Configuration Outputs

The list of options selected by your user during the configuration session is contained in the `<config_outputs>` subelement of `<terminate>`. Each option is enclosed in `<option>` tags and contains the elements described in this section.

Example 4–2 shows an example of configuration outputs in the termination message, with typographical emphasis and comments added.

Example 4–2 Configuration outputs in the termination message

```
<terminate>
  <!-- configuration status goes here -->
  <config_outputs>
    <option>
      <selection_line_id>1846</selection_line_id>
      <parent_line_id>1847</parent_line_id>
      <component_code>143-1490</component_code>
      <quantity>1</quantity>
      <list_price>0.00</list_price>
      <inventory_item_id>1490</inventory_item_id>
      <organization_id>204</organization_id>
      <uom>Ea</uom>
      <discounted_price>0.00</discounted_price>
      <atp_date></atp_date>
    </option>
```

```
    <!-- more options go here -->
  </config_outputs>
  <!-- configuration messages go here -->
</terminate>
```

Subelements

selection_line_id
Contains the ID of the configuration line. It is the same as CZ_CONFIG_ITEMS.CONFIG_ITEM_ID in the Oracle Configurator schema.

component_code
Contains a value extracted from BOM_EXPLOSIONS.COMPONENT_CODE.

parent_line_id
Contains the value from CZ_CONFIG_ITEMS.CONFIG_ITEM_ID for the parent node. It will be "0" for the root.

quantity
Contains the selected quantity for the option.

inventory_item_id
Contains the ID for the item, extracted from MTL_SYSTEM_ITEMS.INVENTORY_ITEM_ID.

organization_id
Contains the organization ID for the item, extracted from MTL_SYSTEM_ITEMS.ORGANIZATION_ID

uom
Contains the unit of measure.

atp_date
Contains the ATP date. This is calculated by using the ATP procedure specified in the initialization message. See ATP Parameters on page 3-13, and Chapter 6, "Integration: Pricing and ATP" on page 6-1.

list_price
Contains the list price for the selected option. This is calculated by using the pricing procedure specified in the initialization message. See Pricing Parameters on page 3-11, and Chapter 6, "Integration: Pricing and ATP" on page 6-1.

discounted_price
Contains the discounted price for the selected option. This is calculated by using the pricing procedure specified in the initialization message. See Pricing Parameters on page 3-11, and Chapter 6, "Integration: Pricing and ATP" on page 6-1.

Configuration Messages

The messages generated by the UI Servlet in response to selections made by your user during the configuration session are contained in the `<config_messages>` subelement of <terminate>. Each message is enclosed in `<message>` tags and contains the elements described in this section.

If there were validation failures during your user's configuration session, each failure on the list of the validation failure objects is returned as a `<message>` element describing the failure. Information about the failure is returned to the UI Servlet as an object of type `oracle.apps.cz.cio.ValidationFailure`, which you can access through the Oracle Configuration Interface Object (CIO); see the Oracle CIO Help for details.

Example 4–3 shows an example of a configuration message in the termination message, with typographical emphasis and comments added.

Example 4–3 Configuration messages in the termination message

```
<terminate>
  <!-- configuration status goes here -->
  <!-- configuration outputs go here -->

  <config_messages>
    <message>
      <message_type>error</message_type>
      <message_text>Config header does not exist in database.</message_text>
    </message>
    <!-- more messages go here -->
  </config_messages>

</terminate>
```

Subelements

component_code
ps_node_id
If present, one of these elements contains the identifier of the option to which this message is related. May be absent, if the message was not generated by a node.

item_name
Contains the name of the option to which this message is related;

message_type
Contains the severity level of the message; possible values are “suggestion”, “warning”, “overridable error”, “error”, “autoselection”, “autoexclusion”, and “not satisfied”.

message_text
Contains the text of the message.

Cancellation

Cancellation occurs after your user closes the configuration window by clicking the Cancel button. Control is returned to the host application, and no configuration information is returned. The termination message contains only the `<exit>` subelement, with a value of "cancel":

```
<terminate>
  <exit>cancel</exit>
</terminate>
```

Error

Error occurs after some condition prevents initialization of the configuration window, or submission of the user’s selections. Such conditions might include:

- Incorrect database connection or user login parameters (see Connection Parameters on page 3-8)
- Lack of any configuration parameters (see Configuration Identification Parameters on page 3-8)
- Incorrect type for a parameter

If there were validation failures during your user’s configuration session, each failure on the list of the validation failure objects is returned as a `<message>`

element describing the failure. Information about the failure is returned to the UI Servlet as an object of type `oracle.apps.cz.cio.ValidationFailure`, which you can access through the Oracle Configuration Interface Object (CIO); see the Oracle CIO Help for details.

Control is returned to the host application, and no configuration information is returned. Any validation failures are returned as messages in the `<config_messages>` element. The termination message contains the `<exit>` subelement, with a value of "error":

```
<terminate>
  <exit>error</exit>
</terminate>
```

The Return URL

The Java servlet specified by the return URL initialization parameter (`return_url`) determines how your host application will use the configuration information produced by your user's selections during a session in the configuration window. The return URL servlet (or just "return URL") is executed upon termination of a configuration session, if you have specified the return URL in your initialization message for the configuration window.

The termination message is passed to the return URL as the value of the XMLmsg argument. The initialization message that was passed to the configurator is also passed to the return URL, as the value of the INITmsg parameter.

The return URL must perform all middle-tier and database processing of the configuration and then return HTML that will close the configuration window and continue with the program flow for the host application.

Specifying the Return URL

You specify the identity of your return URL in the XML initialization message, as the value of the parameter `return_url`:

```
<param
name="return_url">http://www.mysite.com:10130/configurator/Checkout</param>
```

The example parameter above comes from Example 3–3, "Minimal HTML for invoking the configuration window" on page 3-6.

See also:

- Return URL Parameter on page 3-10

- return_url on page 3-19
- Parameter Syntax on page 3-4

Implementing the Return URL

See Example A–1 in Appendix A for an example of a return URL servlet. You can modify this servlet code for the needs of your host application.

In order to use some of the configuration information returned in the termination message (for instance, the outputs described in Configuration Outputs on page 4-5), you can write a method that obtains the value of an element in the termination message by using the `getTagValue()` method of the Checkout servlet.

5

Integration: Batch Validation

How it Works

Batch validation allows a host application to validate a configuration in the background, when the application needs to check whether a configuration is valid.

Batch validation can be called either by:

- passing the batch validation message to the URL of the UI Servlet
- calling the CZ_CF_API.VALIDATE PL/SQL procedure

If the host application is written in PL/SQL, it should call the VALIDATE procedure. There are restrictions in the way that PL/SQL can request data from a URL that require PL/SQL programs to use the cz_cfg_api.validate procedure.

Passing the Batch Validation Message

A batch validation message consists of information defining the configuration context (such as an identifier for the configured model) and a list of configured options. The message can be used to revalidate a previously saved configuration.

A `<batch_validate>` element is composed of an `<initialize>` subelement and a `<config_inputs>` subelement.

The `<initialize>` element is described in Chapter 3, "Integration: Session Initialization". The parameters of the initialization message are described in Initialization Parameter Descriptions on page 3-14. See the database_id parameter description for connectivity information.

The `<config_inputs>` element consists of a list of `<option>` elements, where the `<option>` element is described in Chapter 4, "Integration: Session Termination".

When an <option> element is used in a <config_inputs> element, only the <component_code> and <quantity> elements of the <option> will be used.

Example 5–1 Example of batch validation message

```
<batch_validate>
  <initialize>
    <param name="context_org_id">204</param>
    <param name="config_creation_date">1999/11/23</param>
    <param name="calling_application_id">300</param>
    <param name="responsibility_id">20559</param>
    <param name="config_header_id">21361</param>
    <param name="config_rev_nbr">1</param>
    <param name="read_only">FALSE</param>
    <param name="save_config_behavior">new_revision</param>
    <param name="database_id">ap115sun_dev115</param>
  </initialize>
  <config_inputs>
    <option>
      <component_code>143-1490-1494</component_code>
      <quantity>1</quantity>
    </option>
    <option>
      <component_code>143-297</component_code>
      <quantity>1</quantity>
    </option>
  </config_inputs>
</batch_validate>
```

Calling the CZ_CF_API.VALIDATE Procedure

Table 5–1 describes the parameters for the CZ_CF_API.VALIDATE procedure.

Table 5–1 Parameters for the CZ_CF_API.VALIDATE procedure

Parameter	Data Type	In/Out	Note
config_input_list	CFG_INPUT_LIST	IN	A table of selections to change. the table contains a series of INPUT_ SELECTION records, where each INPUT_SELECTION contains the fields: COMPONENT_CODE of type Varchar2(1200) QUANTITY of type Number
init_message	VARCHAR2	IN	The initialization message.
config_messages	CFG_OUTPUT_PIECES	OUT	The result of the batch validation message. This data type is actually just UTL_HTTP.HTML_PIECES, a table of type Varchar2(2000).
validation_status	Number	OUT	0 if the call succeeds and config_ messages contains a terminate message. Possible values that can be returned are: 0 CONFIG_PROCESSED 1 CONFIG_PROCESSED_NO_ TERMINATE 2 INIT_TOO_LONG 3 INVALID_OPTION_REQUEST 4 CONFIG_EXCEPTION 5 DATABASE_ERROR 6 UTL_HTTP_INIT_FAILED 7 UTL_HTTP_REQUEST_FAILED
URL	VARCHAR2	IN	Defaulted to be FND_ PROFILE.Value('CZ_UIMGR_URL')

Example 5–2 shows fragments from a PL/SQL program that calls the CZ_CF. _ API.VALIDATE procedure

Example 5–2 Example of Calling the CZ_CF. _API.VALIDATE Procedure

```
...
```

```
/*-------------------------------------------------------------------------
Procedure Name : Send_input_XML
Description    : sends the xml batch validation message to SPC that has
                 options that are newly inserted/updated/deleted
                 from the model.
-------------------------------------------------------------------------*/

PROCEDURE Send_input_XML
           ( p_model_line_id       IN NUMBER ,
             p_org_id              IN NUMBER ,
             p_model_id            IN NUMBER ,
             p_config_header_id    IN NUMBER ,
             p_config_rev_nbr      IN NUMBER ,
             p_model_qty           IN NUMBER ,
             p_creation_date       IN DATE ,
             p_deleted_options_tbl IN   OE_Order_PUB.request_tbl_type
                                   := OE_Order_Pub.G_MISS_REQUEST_TBL,
             p_updated_options_tbl IN   OE_Order_PUB.request_tbl_type
                                   := OE_Order_Pub.G_MISS_REQUEST_TBL,
             x_out_XML_msg         OUT LONG ,
             x_return_status       OUT VARCHAR2 )
...
      l_html_pieces                 CZ_CF_API.CFG_OUTPUT_PIECES;
      l_option                      CZ_CF_API.INPUT_SELECTION;
      l_batch_val_tbl               CZ_CF_API.CFG_INPUT_LIST;
...
       Create_hdr_XML
           ( p_model_line_id       => p_model_line_id ,
             p_org_id              => p_org_id ,
             p_model_id            => p_model_id ,
             p_config_header_id    => p_config_header_id ,
             p_config_rev_nbr      => p_config_rev_nbr ,
             p_model_qty           => p_model_qty ,
             p_creation_date       => p_creation_date ,
             x_XML_hdr             => l_XML_hdr);


...
CZ_CF_API.Validate( config_input_list => l_batch_val_tbl ,
                        init_message      => l_XML_hdr ,
                        config_messages   => l_html_pieces ,
                        validation_status => l_validation_status ,
                        URL               => l_url );
```

6

Integration: Pricing and ATP

How it Works

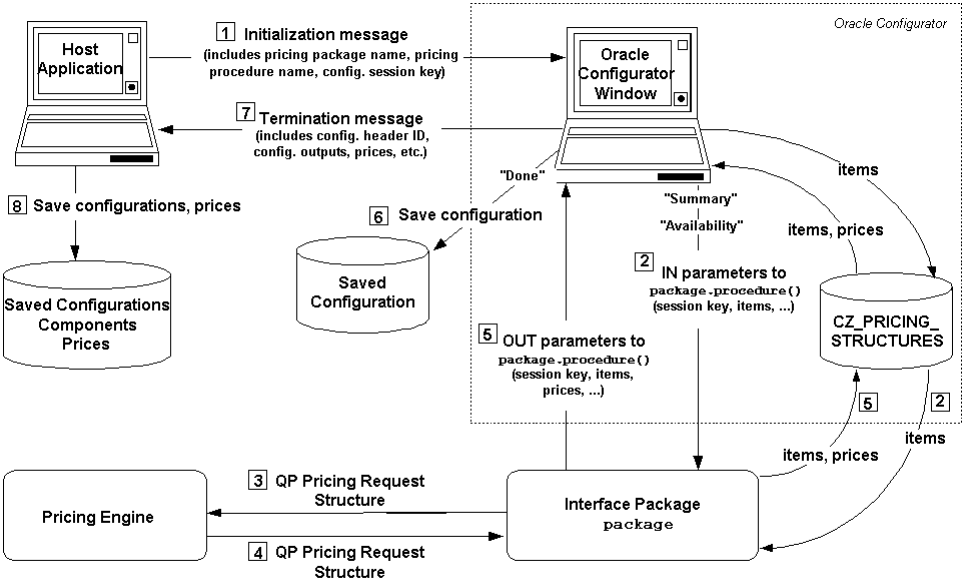
Oracle Configurator allows your host application to display pricing and ATP (Available To Promise) information in the OC configuration window. There is more than one method for presenting this information, to account for these factors:

- The OC configuration window can be called from a variety of different host applications, both Oracle (such as iStore) and non-Oracle.
- Pricing and ATP data can be obtained from either an Oracle Applications 10.7/11.0 database, or an Oracle Applications Release 11*i* database.
- Pricing procedures can be accessed from the Oracle Advanced Pricing engine (QP), which has a highly configurable interface.

To accommodate this flexibility, there is no base pricing inherent in Oracle Configurator, and Oracle Configurator does not call the host application's pricing and ATP procedures directly. Instead, Oracle Configurator defines PL/SQL interfaces for pricing and ATP procedures. The host application is responsible for implementing the "callback" pricing and ATP procedures that implement these interfaces. Parameters that identify these procedures are passed to the OC configuration window at initialization; the OC UI Servlet then calls back to the PL/SQL procedures.

Figure 6–1 shows the architecture for the pricing mechanism in Oracle Configurator. (The architecture for ATP is analogous.)

Figure 6–1 Pricing Architecture in Oracle Configurator



Note: For a fuller description of the pricing architecture for Oracle Configurator, consult the *Oracle Configurator and SellingPoint Administration Guide*.

What You Do

Integrating the configuration window with your pricing or ATP implementation consists primarily of causing your host application (e.g., through the coding of the "Configure" button) to post the XML initialization message to the UI Servlet, passing as initialization parameters the names of your packages and procedures.

To use the OC pricing and ATP interfaces, you must:

1. Install the OC interface packages in your database, by running the appropriate scripts.

   See Database Compatibility on page 6-3 and Installation Scripts on page 6-4.

2. Write your own PL/SQL pricing or ATP procedures, using the OC interfaces. See the *Oracle Configurator and SellingPoint Administration Guide*, and Implementation Scripts on page 6-4 for details.

   See Examples of Pricing and ATP Callback Procedures on page B-1 for examples.

3. Install your packages containing your procedures into the Oracle Applications database.

   You can interface to the Oracle QP pricing engine from your own procedures.

4. In the initialization message that your host application passes to the OC UI Servlet, provide parameters that specify the name of the pricing package, the procedure to use, and the type of pricing to perform.

   See Initialization Parameters on page 6-11 for an example. See Pricing Parameters on page 3-11 and ATP Parameters on page 3-13 for explanation of the parameters.

Database Compatibility

Oracle Configurator works natively with Oracle Applications Release 11*i*, using the Oracle8*i* Enterprise Edition (Release 8.1.x) database.

In order to obtain pricing data from an Oracle8 Enterprise Edition (Release 8.0.x) database, as used with Oracle Applications 10.7/11.0, you must run a "direct import" script. See Installation Scripts on page 6-4 and the *Oracle Configurator and SellingPoint Administration Guide*.

There are several likely scenarios for pricing and ATP integration:

To Integrate with...	You would...
Oracle Applications Release 11*i* database	You write your own callback procedures (which can call the QP Advanced Pricing engine). To import BOM data to the Oracle Configurator schema tables, you run concurrent programs in the Oracle Bills Of Material application. To export orders to Order Management (Oracle Applications Release 11*i*), you use existing or new programming in your host application.

To Integrate with...	You would...
Oracle Applications 10.7/11.0 database	You cannot use the QP Advanced Pricing engine. You must use a BOM model to perform pricing. To import BOM data to the Oracle Configurator schema tables, you run "direct import" scripts. To export orders to Order Entry (Oracle Applications Release 10.7/11.0), you write custom programs, using the SO_INTERFACE_* tables. In OC, you pass the price_list_id parameter in your initialization message to invoke pricing.
Third-party database	For both import and export of pricing data, you must write custom programs.

You can use the callback interface in all these scenarios.

Installation Scripts

These are the scripts that *install* the implementation of the Oracle Configurator pricing and ATP interfaces. Consult the *Oracle Configurator and SellingPoint Administration Guide* for details on when and how to run them. These scripts run certain of the scripts listed in Implementation Scripts on page 6-4.

- DBAdmin\GO_IMPORT.sql
- DBAdmin\InstAppsIntegrate.sql
- DBAdmin\InstAppsIntegrateViaLink.sql

Implementation Scripts

These are the scripts that must be run to make the Oracle Configurator pricing and ATP interfaces available.

> **Note:** You do not normally run these scripts directly. They are run by the scripts listed in Installation Scripts on page 6-4. They are listed here for your information. Consult the *Oracle Configurator and SellingPoint Administration Guide* for details on when and how to run scripts.

These scripts are used with Oracle Applications 10.7/11.0:

- CZ_LIST_PRICE_B.sql
- CZ_LIST_PRICE_S.sql
- CZ_PRC_UTIL_B.sql
- CZ_PRC_UTIL_S.sql
- CZ_ATP_UTIL_B.sql
- CZ_ATP_UTIL_S.sql

These scripts are used with both Oracle Applications 10.7/11.0 and Oracle Applications 11*i*:

- CZ_PRC_CALLBACK_UTIL_B.sql
- CZ_PRC_CALLBACK_UTIL_S.sql
- CZ_ATP_CALLBACK_UTIL_B.sql
- CZ_ATP_CALLBACK_UTIL_S.sql

These scripts are used with Oracle Applications 10.7/11.0 and an Oracle8 Enterprise Edition Release 8.0.x database:

- CZ_PRC_CALLBACK_UTIL_B_80.sql
- CZ_ATP_CALLBACK_UTIL_B_80.sql

The Pricing Callback Interface

The pricing callback interface package provides interfaces for two distinct procedures:

- Price Single Item
- Price Multiple Items

The Price Single Item procedure returns price information for a single item. Table 6–1 describes the parameters for the Price Single Item procedure.

Table 6–1 Price Single Item Procedure Parameters

Parameter	In/Out	Type	Required	Note
`configurator_session_key`	In	Varchar2	Required	Limit of 50 characters
`price_type`	In	Varchar2	Required	Values are: `LIST` or `SELLING`

Table 6–1 (Cont.) Price Single Item Procedure Parameters

Parameter	In/Out	Type	Required	Note
ps_node_id	In	Number	Conditionally Required	Either ps_node_id or item_key is required.
item_key	In	Varchar 2	Conditionally Required	Either ps_node_id or item_key is required.[1]
quantity	In	Number	Required	
uom_code	In	Varchar 2	Required	
list_price	Out	Number	n/a	
selling_price	Out	Number	n/a	This is the unit selling price. OC automatically multiplies this price by quantity to produce the total price (extension) for the item.
msg_data	Out	Varchar 2	n/a	

[1] If the model was imported from an Oracle BOM, then this key is COMPONENT_CODE:EXPLOSION_TYPE:ORGANIZATION_ID:TOP_ITEM_ID.

The Price Multiple Items procedure returns price information for a group of items. Table 6–2 describes the parameters for the Price Multiple Items procedure.

Table 6–2 Price Multiple Items Procedure Parameters

Parameter	In/Out	Type	Required	Note
configurator_session_key	In	Varchar2	Required	
price_type	In	Varchar2	Required	Values are: LIST or SELLING
config_total_price	Out	Number	n/a	

The parameters of the interface are passed by positional notation, so you can name the parameters as desired, as long as you retain the positionality specified in Table 6–1 and Table 6–2.

Use of the Database in the Price Multiple Items Procedure

When you specify the Price Multiple Items procedure, Oracle Configurator stores the list of items to be priced in the database table CZ_PRICING_STRUCTURES. This table is described in Table 6–3 on page 6-7. (See the *Oracle Configurator Technical Reference Manual* for details on Oracle Configurator tables.)

Your pricing package must retrieve the items from this table and call the pricing engine, then capture all of the results and update the CZ_PRICING_STRUCTURES table with list and/or selling prices. Oracle Configurator retrieves the prices from the CZ_PRICING_STRUCTURES table during the configuration session, so that they can be presented in the configuration window. When the configuration window exits, OC deletes the pricing records from the CZ_PRICING_STRUCTURES table.

If your host application needs to retain the prices for use after the end of the current configuration session, then your pricing package must store the results in application-specific tables (or some other location that is insensitive to the Oracle session). Oracle Configurator does not reference the contents of these application-specific tables.

Table 6–3 CZ_PRICING_STRUCTURES Database Table

Column Name	Data Type	Null?	Description
CONFIGURATOR_SESSION_KEY	Varchar2	Not Null	Limit of 50 characters. Primary key. Identifies a configurator session. Only one configuration can be handled in the session.
SEQ_NBR	Number	Not Null	Primary key. Sequence number of the item in the list of items.
PS_NODE_ID	Number		Limit of 9 digits. PS_NODE_ID is a foreign key reference into the CZ_PS_NODES table, which defines the "configuration" identity of the object.
ITEM_KEY	Varchar2	Not Null	Limit of 2000 characters. ORIG_SYS_REF for imported items or PS_NODE_ID for non-imported items.
ITEM_KEY_TYPE	Number	Not Null	Limit of 9 digits. Set to 1^1 if ITEM_KEY is ORIG_SYS_REF. Set to 2^2 if ITEM_KEY is PS_NODE_ID.

Table 6–3 (Cont.) CZ_PRICING_STRUCTURES Database Table

Column Name	Data Type	Null?	Description
QUANTITY	Number		Limit of 9 digits. Item quantity
UOM_CODE	Varchar2		Limit of 3 characters. UOM code
LIST_PRICE	Number		List price
SELLING_PRICE	Number		Selling price
MSG_DATA	Varchar2		Limit of 2000 characters. Message text filled in by your host application.

[1] Value of CZ_PRC_CALLBACK_UTIL.G_ITEM_KEY_BOM_NODE.
[2] Value of CZ_PRC_CALLBACK_UTIL.G_ITEM_KEY_PS_NODE.

Examples of the Pricing Callback Interface

Example 6–1 on page 6-8 shows a possible implementation of the callback interface for both single and multiple-item pricing procedures.

Example 6–3 on page 6-12 shows how you would specify pricing parameters in your initialization message.

Example B–1 on page B-1 provides a minimal example of the use of the callback interface.

Example 6–1 Pricing Callback Interface

```
PACKAGE CZ_PRICE_TEST AUTHID CURRENT_USER AS

 PROCEDURE price_single_item(configurator_session_key IN VARCHAR2,
                             price_type IN VARCHAR2,
                             ps_node_id IN NUMBER,
                             item_key IN VARCHAR2,
                             quantity IN NUMBER,
                             uom_code IN VARCHAR2,
                             list_price OUT NUMBER,
                             selling_price OUT NUMBER,
                             msg_data OUT VARCHAR2);

  PROCEDURE price_multiple_items (p_configurator_session_key IN VARCHAR2,
                                  p_price_type IN VARCHAR2,
```

```
                                        p_total_price OUT NUMBER);

END;
```

The ATP Callback Interface

The "Get ATP Dates" procedure returns availability dates for all selected options and for the configuration as a whole. Table 6–4 describes the parameters for the Get ATP Dates procedure.

Table 6–4 ATP Procedure Parameters

Parameter	In/Out	Type	Required	Note
configurator_session_key	In	Varchar2	Required	Limit of 50 characters
warehouse_id	In	Number	Required	
ship_to_org_id	In	Number	Conditionally Required	You must provide either `ship_to_org_id` (by itself), or both `customer_id` and `customer_site_id`.
customer_id	In	Number	Conditionally Required	
customer_site_id	In	Number	Conditionally Required	
requested_date	In	Date	Required	Defaults to value of SYSDATE.
ship_to_group_date	Out	Date	n/a	

The parameters of the interface are passed by positional notation, so you can name the parameters as desired, as long as you retain the positionality specified in Table 6–4.

Use of the Database with the ATP Callback Interface

When you specify the Get ATP Dates procedure, Oracle Configurator stores the list of items to obtain ATP dates for in the database table CZ_ATP_REQUESTS. This table is described in Table 6–5 on page 6-10. (See the *Oracle Configurator Technical Reference Manual* for details on Oracle Configurator tables.)

Your ATP package must retrieve the items from this table and call the `call_atp()` procedure defined in your ATP package, then capture all of the results and update the CZ_ATP_REQUESTS table with ATP dates.

Oracle Configurator retrieves the ATP dates from the CZ_ATP_REQUESTS table during the configuration session, so that they can be presented in the configuration window. When the configuration window exits, OC deletes the ATP dates from the CZ_ATP_REQUESTS table.

If your host application needs to retain the ATP dates for use after the end of the current configuration session, then your ATP package must store the results in application-specific tables (or some other location that is insensitive to the Oracle session). Oracle Configurator does not reference the contents of these application-specific tables.

Table 6–5 CZ_ATP_REQUESTS Database Table

Column Name	Data Type	Null?	Description
CONFIGURATOR_SESSION_KEY	Varchar2	Not Null	Limit of 50 characters. Primary Key. Identifies a configurator session. Only one configuration can be handled in the session.
SEQ_NO	Number	Not Null	Primary key. Sequence number of the item in the list of items
PS_NODE_ID	Number		Limit of 9 digits. PS_NODE_ID is a foreign key reference into the CZ_PS_NODES table, which defines the "configuration" identity of the object.
ITEM_KEY	Varchar2	Not Null	Limit of 2000 characters. ORIG_SYS_REF for imported items or PS_NODE_ID for non-imported items.
ITEM_KEY_TYPE	Number		Limit of 9 digits. Set to 1^1 if ITEM_KEY is ORIG_SYS_REF. Set to 2^2 if ITEM_KEY is PS_NODE_ID.
QUANTITY	Number		Limit of 9 digits. Item quantity
UOM_CODE	Varchar2		Limit of 3 characters. UOM code
MSG_DATA	Varchar2		Limit of 2000 characters. Message text filled in by your host application.
INV_ORG_ID	Number		Limit of 9 digits.

Table 6–5 (Cont.) CZ_ATP_REQUESTS Database Table

Column Name	Data Type	Null?	Description
SHIP_TO_DATE	Date		

[1] Value of CZ_PRC_CALLBACK_UTIL.G_ITEM_KEY_BOM_NODE.
[2] Value of CZ_PRC_CALLBACK_UTIL.G_ITEM_KEY_PS_NODE.

Examples of the ATP Callback Interface

Example 6–2 on page 6-11 shows an implementation of the callback interface for ATP procedures.

Example 6–3 on page 6-12 shows how you would specify ATP parameters in your initialization message.

Example B–3, "Example of Callback ATP Procedure" on page B-2 provides an example in context.

Example 6–2 ATP Callback Interface

```
PACKAGE cz_atp_callback AS

  PROCEDURE call_atp (p_config_session_key IN VARCHAR2,
                      p_warehouse_id IN NUMBER,
                      p_ship_to_org_id IN NUMBER,
                      p_customer_id IN NUMBER,
                      p_customer_site_id IN NUMBER,
                      p_requested_date IN DATE,
                      p_ship_to_group_date OUT DATE);

END cz_atp_callback;
```

Initialization Parameters

Example 6–3 is a test page that shows how you would specify pricing and ATP parameters in your initialization message. The names of the pricing and ATP parameters are typographically emphasized. This example shows parameters for use with Oracle Applications Release 11*i*. See Pricing Parameters on page 3-11 and ATP Parameters on page 3-13.

Example 6–3 Initialization message using 11i pricing and ATP parameters

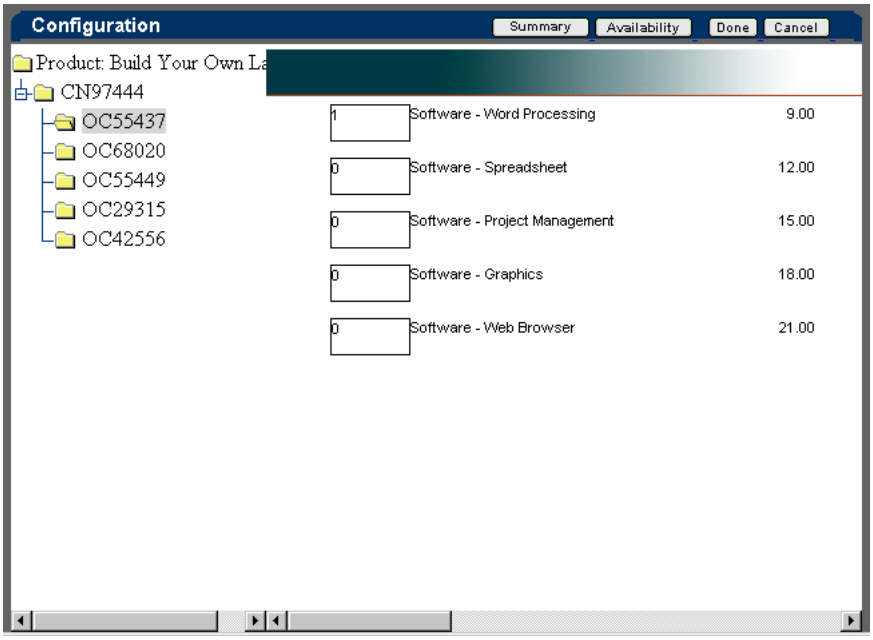
```
<html>
<head>
<title>Project Test</title>
<script language="javascript">
function init() {
  document.test1.submit();
}
</script>
</head>

<body onload="init();">
<form
action="http://www.mysite.com:10130/servlets/oracle.apps.cz.servlet.UiServlet"
method="post" id="test1" name="test1"><input type="hidden" name="XMLmsg"
value='<initialize>
<param name="alt_database_name">jdbc:oracle:thin:@server01:1521:vis11</param>
<param name="user">apps</param>
<param name="pwd">apps</param>
<param name="ui_type">DHTML</param>
<param name="gwyuid">APPLSYSPUB/PUB</param>
<param name="fndnam">APPS</param>

<param name="ui_def_id">3080</param>

<param name="pricing_package_name">cz_price_test</param>
<param name="price_mult_items_proc">price_multiple_items</param>
<param name="price_single_item_proc">price_single_item</param>

<param name="configurator_session_key">1234</param>

<param name="atp_package_name">cz_atp_callback_stub</param>
<param name="get_atp_dates_proc">call_atp</param>
<param name="warehouse_id">207</param>
<param name="customer_id">1000</param>

</initialize>'></form>

<br>Loading configurator...

</body>
</html>
```

In order to obtain the final prices calculated by your pricing package, you need to specify a value of `full` for the initialization parameter terminate_msg_behavior (described on page 3-20). When your configuration session terminates normally, Oracle Configurator returns the final prices in the termination message. Your host application can then save the prices as needed.

Testing Pricing and ATP Integration

Figure 6–2 on page 6-13 shows the effect of the initialization message shown in Example 6–3 on page 6-12.

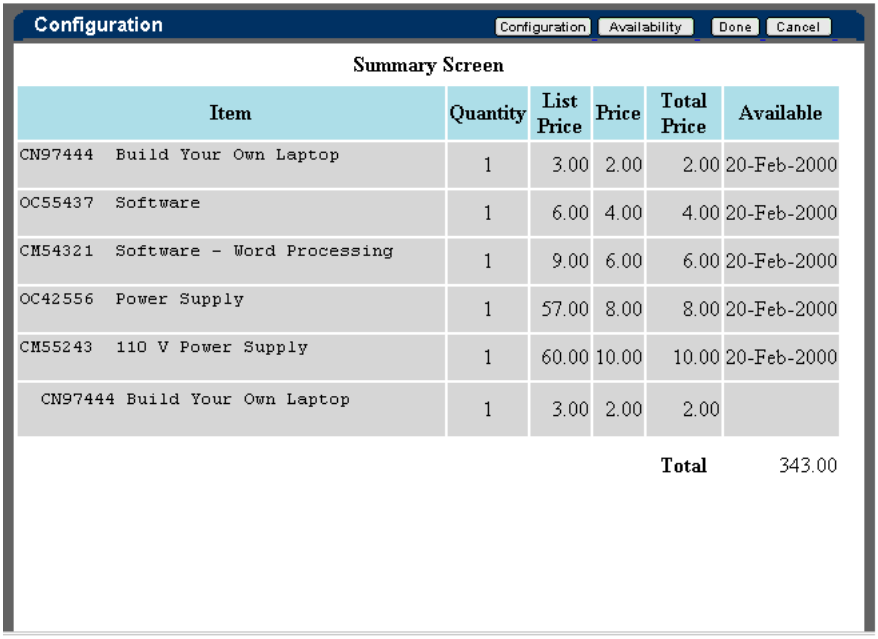
Figure 6–2 Pricing Data in the Configuration Window



In the content pane of the OC configuration window, list prices are displayed next to each item. These are obtained from the database by Oracle Configurator.

Figure 6–3 on page 6-14 shows how pricing and ATP data is displayed when you click the Summary button.

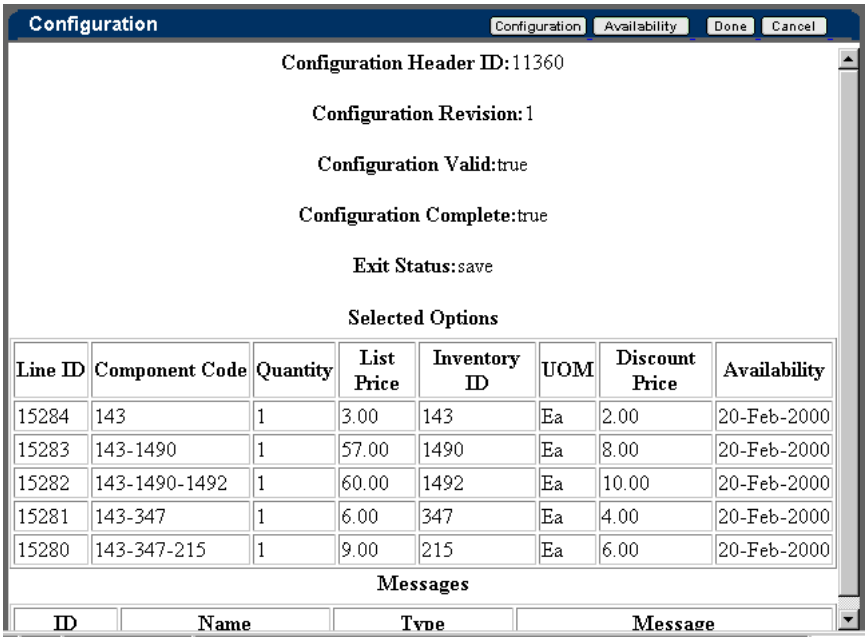
Figure 6–3 Pricing and ATP Data in the Summary Pane



Your pricing procedures are used to calculate the selling prices and total price. If you are using the single-item pricing procedure (specified by the price_single_item_proc parameter), then the total price is calculated by summing up all the selling prices. If you are using the multiple-item pricing procedure (specified by the price_mult_items_proc parameter), then the total price is calculated by your callback procedure.

Figure 6–4 on page 6-15 shows how pricing and ATP data is displayed when you click the Done button. See Submission on page 4-3 for an explanation of the data provided.

Figure 6–4 Pricing and ATP Data in the Submission Display



Configuration

Configuration | Availability | Done | Cancel

Configuration Header ID: 11360

Configuration Revision: 1

Configuration Valid: true

Configuration Complete: true

Exit Status: save

Selected Options

Line ID	Component Code	Quantity	List Price	Inventory ID	UOM	Discount Price	Availability
15284	143	1	3.00	143	Ea	2.00	20-Feb-2000
15283	143-1490	1	57.00	1490	Ea	8.00	20-Feb-2000
15282	143-1490-1492	1	60.00	1492	Ea	10.00	20-Feb-2000
15281	143-347	1	6.00	347	Ea	4.00	20-Feb-2000
15280	143-347-215	1	9.00	215	Ea	6.00	20-Feb-2000

Messages

ID	Name	Type	Message

7

User Interface

Oracle Configurator includes a UI Server that renders a DHTML representation of an Oracle Configurator Model and user interface, as defined in Configurator Developer and published in the Active Model and Active UI. This document does not explain how to customize a custom user interface not generated through Configurator Developer, or using a non-Oracle host application.

Oracle Configurator provides a set of HTML files that make up the parts of a single HTML Template. This Template can be used as a model for producing alternate Template designs. This enables you to incorporate the configuration window into other host applications, retaining the functionality built into the configuration window while adapting its look and feel to match the surrounding frames of the host application.

Custom Template versions can be created in any HTML editor or text editor.

How it Works

The HTML Template has a structure that supports the operation of the configuration window.

Structure of the HTML Template

It is essential to understand the basic structure of the HTML Template, in order to know which parts you can customize, and how.

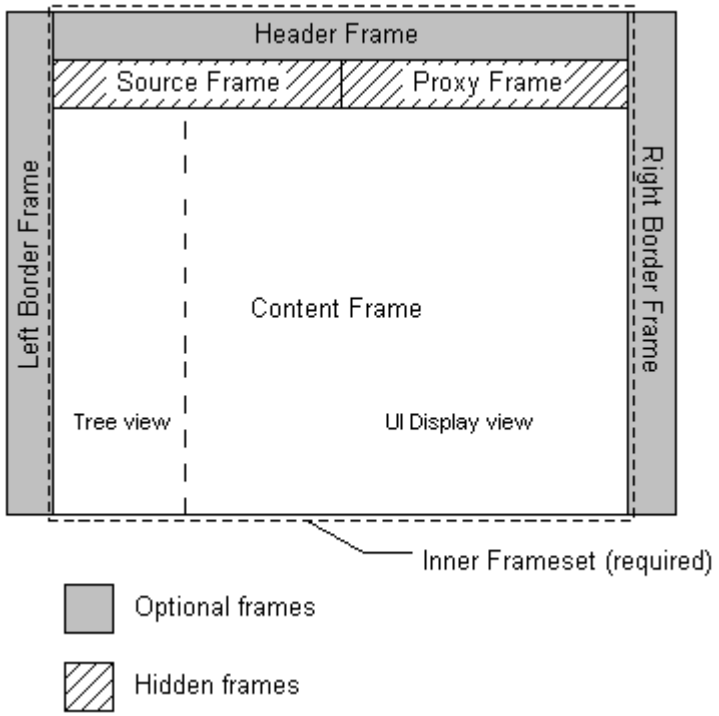
The Template is divided into two areas:

- required template elements, which are essential for supporting the operation of the DHTML components
- optional template elements, which you can customize and augment

The required structure must be incorporated into every custom Template. The structure of the template, as shown in Figure 7–1 on page 7-2, is:

- the optional, customizable Outer Frameset, which consists of:
  - the required Inner Frameset
  - the optional, customizable Left and Right Border Frames
- the required, non-customizable Inner Frameset, which consists of:
  - the optional Header Frame
  - the required Source and Proxy Frames
  - the required Content Frame

Figure 7–1 The Structure of the HTML Template



Note: For the sake of a convenient geometry, the OC template files have defined the Inner Frameset to include the Header Frame. However, it is only the Source Frame,

Proxy Frame and Content Frame that are required for the operation of the OC configuration window.

Table 7–1 lists the constraints on customizing the Template, and the files in which the elements of the Template are implemented.

Table 7–1 Constraints on the HTML Template

Required?	Customize?	Frame/Frameset				File
No	Yes	Outer Frameset				czFraIE.htm czFraNS.htm
No	Yes		Left Border Frame			czLeft.htm
Yes	No		Inner Frameset			czFraIE.htm czFraNS.htm
No	Yes			Header Frame		czHeader.htm
Yes	No			Source and Proxy Frameset		czFraIE.htm czFraNS.htm
Yes	No				Source Frame	czSource.htm
Yes	No				Proxy Frame	Proxy.class
Yes	No			Content Frame		czCntnt.htm
Yes	No			Tree view		cztree.htm
Yes	No			Display view		czdisp.htm
No	Yes		Right Border Frame			czRight.htm

All of the framesets are defined in the files czFraIE.htm and czFraNS.htm.

The Outer Frameset divides the hosting frame into three columns so that you can have a left and right border.

The Content Frame is the location of the configuration window, including the summary. The Source Frame contains the JavaScript controls that define the behavior of the DHTML user interface. The Proxy Frame is the communication center between the configuration window and the UI Server.

There are no controls for submitting and canceling configurations, in this structure. It is up to the implementer to put the controls for triggering functions in the Template or in the host application. Triggering functions are:

- submit
- cancel
- show configuration or summary information
- display ATP information

For an example of how the HTML Template appears when invoked in a browser, see Figure A–1 on page A-6.

The Source and Proxy Frames are hidden from the end user, by setting their height allocations to 0 (zero) in the Inner Frameset, which contains them. The Inner Frameset is defined in the browser-specific frameset files listed in Table 7–2.

Table 7–2 Browser-Specific Frameset Files

Browser	Frameset File	Java System Property
Netscape	czFraNS.htm	cz.uiservlet.templateurl
Internet Explorer	czFraIE.htm	cz.uiservlet.templateurl.ie

If you substitute different frameset files, you need to set the corresponding Java System Properties accordingly. For information on how to set these parameters, see the reference section on Java System Property Parameters for the UI Servlet in the *Oracle Configurator Installation Guide*. You can also specify the frameset file in the initialization message for the UI Servlet, using the parameter template_url on page 3-20.

The Header Frame

This frame loads the HTML page **czHeader.htm** (Configurator Header). It consists of your choice of GIFs to display the desired header graphics, and a set of image buttons, listed in Table 7–3 on page 7-5.

Figure 7–2 The czHeader frame

Table 7–3 Buttons in the Header Frame

Button Name	Description
Summary/Configuration	This button is scripted to toggle the Contents Frame between the Oracle Configurator UI and the Summary screen. The script also changes the caption on the button to read "Summary" when the Oracle Configurator UI is displayed and "Configuration" when the Summary screen is displayed.
Availability	This button is scripted to populate the current display with ATP dates. When triggered for the Summary screen, it refreshes the entire Summary page with data that includes the ATP information.
Done	This button is scripted to save the configuration, terminate the session, and return control to the host application.
Cancel	This button is scripted to terminate the session and return control to the host application without saving the configuration.

If you need to add your own buttons, you can add functions to those defined in czHeader.htm. Do not modify the existing functions, or the configuration window may not operate correctly.

As a convenience, the complete source code for czHeader.htm is provided in Example C–1 on page C-1.

The Left and Right Border Frames

These frames load the HTML pages **czLeft.htm** and **czRight.htm**, respectively. They each display only a background color. You can add images and controls as required for your host application.

What You Do

- You can customize the decorations in the Header frame, and perhaps in the Left and Right frames.
- You can customize the buttons in the Header frame.
- You can add new buttons in the Header frame.
- You do *not* modify the Content Frame, since its contents are automatically generated by the UI Servlet.

The Default Configuration Window

You normally do not have to define a user interface for the configuration window; a default one may be already available.

The layout, navigational model, controls, and other objects displayed in the configuration window are derived from a User Interface definition in the Oracle Configurator schema. The User Interface definition is created from a Model structure in the Oracle Configurator Developer. The Oracle Configurator window can be further customized in Configurator Developer.

The Oracle Configurator window displayed in the your web browser presents a configuration Model. Once selections have been made by your user, the configuration window presents the resulting configuration for that Model. Unsatisfied selections are indicated by a customizable icon. By default, the icon is a red asterisk.

Customizing the User Interface in Oracle Configurator Developer

You generate and modify the default UI in Oracle Configurator Developer. Your design is displayed in the UI display view of the Content Frame.

If you want to call Functional Companions from your user interface, define controls to call them, in Configurator Developer.

See the *Oracle Configurator Developer Tutorial* and Oracle Configurator Developer User's Guide.

Restrictions on the Configuration Window

- You cannot use BMP (Windows bitmap) files in your user interface for the configuration window, since this file format is not compatible with rendering in a DHTML context. This will affect you if the User Interface that you defined in Oracle Configurator Developer included any BMP files.

  The configuration window can use GIF, JPG, and other formats compatible with web browsers. The User Interface defined in Oracle Configurator Developer can include BMP files if it is only used with the Oracle Configurator in the Oracle SellingPoint application.

Customizing the HTML Templates

The template that you are most likely to modify is the Header Frame. For the HTML of this frame, see Example C–1, "The Header Frame template file (czHeader.htm)" on page C-1.

You may also want to modify the frameset files for the Netscape and Internet Explorer browsers (czFraNS.htm and czFraIE.htm, respectively). See Table 7–2 on page 7-4.

Warning: If you modify any of the template files, be sure to make backups. The template files are overwritten whenever you reinstall Oracle Configurator.

The modifiable template files are commented for your guidance.

Specifying the Location of Media Files

When you design a User Interface in Configurator Developer, you can specify the location of custom image media files for your host application: icons, backgrounds, button shapes, and the like. When your host application uses the DHTML configuration window, the UI Server strips the path information (which is stored in the database) from the name of these media files, and uses only the filename itself.

The OC HTML template files set a global variable IMAGESPATH, pointing to the location of the media files, that is read by the JavaScript controls that populate the configuration window. By default, IMAGESPATH is set to /OA_MEDIA/, which is located under the document root directory for your servlet. The web server serves HTML documents from the document root directory, unless it is configured to search elsewhere.

Oracle Configurator stores its own standard media files in OA_MEDIA. Here is a fragment from Example C–1, "The Header Frame template file (czHeader.htm)" on page C-1, showing the location of an image file for the Done button (emphasized):

```
<!-- done button -->
    <a href="javascript:doneClick();">
      <img name="IMG-done" src="/OA_MEDIA/czDone0.gif" border="0" WIDTH="37"
HEIGHT="15">
    </a>
```

If you have custom media files to use, you can store them in OA_MEDIA. If you wish to use a different location, it is recommended that you add a directory under OA_MEDIA, then modify certain HTML template files to reflect that location.

1. Change this code in czSource.htm:

```
//Modify it depending on your server installation directory.
//** Do not forget to put the trailing SLASH at the end
var IMAGESPATH = '/OA_MEDIA/alt/';
var SOURCEPATH = '/OA_HTML/';
```

2. Change this code in czHeader.htm:

```
/* Global string storing physical or logical path to server
     graphics. */
  var IMAGESPATH = '/OA_MEDIA/alt/';
```

3. Copy your custom media files, and the OC media files, to the new directory.

Hiding the Model Tree

By default, the Content Frame displays a tree view of the configuration Model next to a view of the User Interface that was defined in Configurator Developer. If you want to hide the tree view, you can do so by setting a Java System Property Parameter to be passed to the UI Servlet. The parameter is `cz.frameset.allocations.top`. Its default setting is:

```
cz.frameset.allocations.top=30%,*
```

To hide the tree, set the width of the tree view to 0 (zero):

```
cz.frameset.allocations.top=0,*
```

For information on how to set this parameter, see the reference section on Java System Property Parameters for the UI Servlet in the *Oracle Configurator and SellingPoint Administration Guide*

Modifying the Header Bar

As a simple example of the modifications you can make directly to the HTML template files, try changing the text of the header bar of the configuration window.

Locate the following lines in /OA_HTML/US/czHeader.htm:

```
<!-- label -->
<div id="DIV-label" nowrap>
```

```
    <font face="Arial" pointsize="12pt" color="white">
      <b>Configuration</b>
    </font>
  </div>
```

The typographical emphasis indicates what you will change. This HTML file is provided for convenient reference as Example C–1, "The Header Frame template file (czHeader.htm)" on page C-1.

Now change the label text as shown:

```
<b>My Configuration Window</b>
```

The changes will take effect when you refresh the web browser displaying the UI Servlet.

A

The Checkout Servlet

Example A–1 is the complete source code for Checkout.java, which you can use as a template for constructing your own return URL servlet.

The Java servlet shown here obtains the value of the `valid_configuration` element from the configuration outputs element of the termination message and displays it in an HTML frame that takes the place of the configuration window after your user has closed the window and saved the results of the configuration session.

See the following sections for background.

The Return URL	on page 4-9
Integration: Session Termination	on page 4-1
Submission	on page 4-3
Configuration Outputs	on page 4-5
valid_configuration	on page 4-4

The parts of the code that you should customize to work with a different configuration output element than `valid_configuration` are typographically emphasized.

Example A–1 The default Checkout servlet (Checkout.java)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

import oracle.apps.cz.common.XmlUtil;
import oracle.xml.parser.v2.XMLDocument;
import org.xml.sax.SAXException;
```

```
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class Checkout extends HttpServlet {

  // Responds to the UiServlet request containing the <terminate> XML message
  public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    String terminateString = request.getParameter("XMLmsg");
    XMLDocument terminateDoc;
    try {
      terminateDoc = XmlUtil.parseXmlString(terminateString);
    } catch (SAXException se) {
      throw new ServletException(se.getMessage());
    }
    String validConfig = getValidConfig(terminateDoc);
    System.err.println("configuration valid?: " + validConfig);

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<script language=\"javascript\">");
    out.println("parent.location = \"/configurator/Checkout?ValidConfig=" + validConfig + "\"");
    out.println("</script>");
    out.println("</html>");
  }

  // Responds to the secondary request for the page to replace the content frame
  // containing the ValidConfig
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    String validConfig = request.getParameter("ValidConfig");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>Checked Out with Valid Configuration</title></head>");
    out.println("<body>");
    out.println("Configuration Valid?: " + validConfig);
    out.println("</body>");
    out.println("</html>");
  }

  String getValidConfig(XMLDocument doc) {
    return getTagValue(doc, "valid_configuration", null); // get element from termination msg
  }
```

```
  String getTagValue(XMLDocument doc, String tagName, String defaultValue) {
    Node n = doc.getDocumentElement();
    if (n != null) {
      NodeList nl = n.getChildNodes();
      if (nl != null) {
        for (int i = 0; i < nl.getLength(); i++) {
          Node cn = nl.item(i);
          if (cn.getNodeName().equals(tagName)) {
            NodeList cnl = cn.getChildNodes();
            if (cnl != null) {
              return cnl.item(0).getNodeValue();
            }
          }
        }
      }
    }
    return defaultValue;
  }
}
```

B

Examples of Pricing and ATP Callback Procedures

This appendix contains minimal examples of PL/SQL procedures you might write to use the OC callback interface for pricing and ATP procedures.

See the following sections for background:

Integration: Pricing and ATP on page 6-1

The Pricing Callback Interface on page 6-5

The ATP Callback Interface on page 6-9

Pricing Parameters on page 3-11

ATP Parameters on page 3-13

Example B–1 Example of Single-item Callback Pricing Procedure

```
PROCEDURE price_single_item (configurator_session_key IN VARCHAR2,
                             price_type IN VARCHAR2,
                             ps_node_id IN NUMBER,
                             item_key IN VARCHAR2,
                             quantity IN NUMBER,
                             uom_code IN VARCHAR2,
                             list_price OUT NUMBER,
                             selling_price OUT NUMBER,
                             msg_data OUT VARCHAR2) AS
BEGIN

  IF item_key IS NULL THEN
  -- hard-coded price amounts
    list_price := 2.0;
```

```
    selling_price := 2.0;
  ELSE
    list_price := 1.0;
    selling_price := 1.0;
  END IF;

  -- debugging information
  msg_data := configurator_session_key || ' : ' || price_type || ' : ' ||
              item_key || ' : ' || uom_code;

END price_single_item;
```

Example B–2 Example of Multiple-item Callback Pricing Procedure

```
PROCEDURE price_multiple_items (p_configurator_session_key IN VARCHAR2,
                                p_price_type IN VARCHAR2,
                                p_total_price OUT NUMBER) AS
BEGIN
  update cz_pricing_structures set list_price = 3.0*seq_nbr,
    selling_price = 2.0*seq_nbr,
    where configurator_session_key =
    p_configurator_session_key;

-- calculation using pricing table for storage
    select sum(selling_price) into p_total_price from cz_pricing_structures
    where configurator_session_key = p_configurator_session_key;

  -- hard-coded price amount
  -- p_total_price := 343.00;

END price_multiple_items;
```

Example B–3 Example of Callback ATP Procedure

```
  PROCEDURE call_atp (p_config_session_key IN VARCHAR2,
                      p_warehouse_id IN NUMBER,
                      p_ship_to_org_id IN NUMBER,
                      p_customer_id IN NUMBER,
                      p_customer_site_id IN NUMBER,
                      p_requested_date IN DATE,
                      p_ship_to_group_date OUT DATE) IS
  BEGIN
```

```
  update cz_atp_requests set ship_to_date = sysdate-10
    where configurator_session_key
    = p_config_session_key;

  p_ship_to_group_date := sysdate;
END call_atp;
```

C

The Header HTML Template

Example C–1 is the complete source code for the file **`czHeader.htm`**, which is provided with Oracle Configurator as a template for constructing your own header frame for the configuration window.

See the following section for background.

The Header Frame on page 7-4

Example C–1 The Header Frame template file (czHeader.htm)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<head>
<title></title>

<!-- Define Cascading Style Sheet (CSS) entries to enabled absolute     positioning of the
document elements.          (in order of style sheet appearance...)     - left-side tab (corner)
of the header background     - center tab stretched across frame on which the buttons are
rendered     - right-side tab or corner     - caption     - button layer which contains the
function-bound image layers-->

<style type="text/css">
  #DIV-tableft   {position:absolute ; left: 0;   top: 4; width: 7;   height: 21}
  #DIV-tabcenter {position:absolute ; left: 7;   top: 2; width: 624; height: 21}
  #DIV-tabright  {position:absolute ; left: 631; top: 4; width: 7;   height: 21}
  #DIV-label     {position:absolute ; left: 15;  top: 4; width: 23;  height: 21}
  #DIV-buttons   {position:absolute ; left: 360;  top: 3; width: 270; height: 21}
</style>

<script language="JavaScript">
<!--// begin browser no-script spoof
```

```
  /* Define global boolean values to signal browser type to other
     functions on this page. */
var ns4 = (document.layers)? true : false;
var ie4 = (document.all)? true : false;

  /* Global string storing physical or logical path to server
     graphics. */
  var IMAGESPATH = '/OA_MEDIA/';

  /* Time in milleseconds for the page to wait before resetting
     button images. */
  var ROLL_RESET_TIMEOUT = 2000;

  /* Enumeration of button image element indices...  these are
     needed to access the images through the Netscape DOM.

     Add more constants here according to the layout of your page. */
  var BTNIX_VIEW = 0;
  var BTNIX_ATP = 1;
  var BTNIX_DONE = 2;
  var BTNIX_CANCEL = 3;

  /* Flag to indicate that this docuement cannot be written to
     by the UiBroker's publish methods. */
var isRenderable = false;


  /* The following code hadles browser resize for Netscape browsers */
  function handleResize() {
    location.reload();
    return false;
  }

  if (ns4) {
    window.captureEvents(Event.RESIZE);
    window.onresize = handleResize;
  }

//--> end browser no-script spoof
</script>

<script language="JavaScript">
  var ub = null; // stores local reference to the uibroker object

  /** Funtion getUiBroker returns a pointer to the uibroker object
```

```
   * which is 'always' located in the czSrc frame. If this reference is
   * invalid due to customization,  this function must be modified.
   *
   * A null return value indicates that there was no instance to be found.
   * If this occurs a timeout and retry is recommended. It's likely that
   * on load, the browser has completed downloading this frame first.
   */
  function getUiBroker()
  {
    if (ub == null) {
      // make sure the czSrc frame has been loaded
      if (parent.frames.czSrc.isLoaded) {
        ub = parent.frames.czSrc.getUiBroker();
      }
    }
    if (ub == null) {
       //log that the UiBroker has not been found. Some sort of internal error!!
    }
    return ub;
  }

  /** Function viewClick handles a 'mouseclick' on the view button.
    * By default this toggles between the configuration and summary
    * screens by requesting that the server publish to the content
    * frame
    */
  function viewClick()
  {
    var uBroker = this.getUiBroker();

    if (uBroker != null) {
      //setting the reference of this page in uibroker for futher reference
      uBroker.hdr_ref = this;

      // check the current state of the content frame
      // through the uibroker's method (function).
      if (uBroker.getContentMode() == 'config') {
        // ask the server to publish the summary screen.
    uBroker.showSummary();
        setImage('IMG-view', BTNIX_VIEW, IMAGESPATH + 'czConf0.gif');
        // roll the current image
        rollImage('IMG-view', BTNIX_VIEW, IMAGESPATH + 'czConf1.gif', IMAGESPATH +
'czConf0.gif');
      }
      else {
```

```
        // ask the server to publish the configuration screen.
                        if (uBroker.pageHistory.peek () ==  '<show-summary/>')
                        {
                           uBroker.pageHistory.pop();
                        }
 uBroker.showConfiguration();
        setImage('IMG-view', BTNIX_VIEW, IMAGESPATH + 'czSumm0.gif');
        // roll the current image
        rollImage('IMG-view', BTNIX_VIEW, IMAGESPATH + 'czSumm1.gif', IMAGESPATH +
'czSumm0.gif');
}
    }
    else {
      // wait for the uibroker.
    }
  }

  /** Function doneClick handles a 'mouseclick' on the done button.
    * By default this initiates a uibroker function call,  which, in turn,
    * communicates to the server that the configuration session is complete.
    *
    * Code may be added here to signal a containing frame that the configuration
    * is complete.
    */
  function doneClick()
  {
    var uBroker = this.getUiBroker();

    if (uBroker != null) {
      // tell the server we are done.
  uBroker.completeConfiguration();
      rollImage('IMG-done', BTNIX_DONE, IMAGESPATH + 'czDone1.gif', IMAGESPATH + 'czDone0.gif');
    } else {
      // wait for the uibroker.
    }
  }

  /** Function cancelClick handles a 'mouseclick' on the cancel button.
    * By default this initiates a uibroker function call,  which, in turn,
    * communicates to the server that the configuration session is canceled.
    *
    * Code may be added here to signal a containing frame that the configuration
    * is complete.
    */
  function cancelClick()
```

```
  {
    var uBroker = this.getUiBroker();

    if (uBroker != null) {
      // tell the server we are done.
  uBroker.cancelConfiguration();
      rollImage('IMG-cancel', BTNIX_CANCEL, IMAGESPATH + 'czCanc1.gif', IMAGESPATH +
'czCanc0.gif');
    } else {
      // wait for the uibroker.
    }
  }

  /** Function atpClick handles a 'mouseclick' on the atp button.
    * By default this initiates a uibroker function call,  which, in turn,
    * communicates to the server that Availability to Promise should be displayed
    * with items in the interactive controls (lists/checkboxes/etc.) located in the
    * czDisp - display frame.
    *
    * Code may be added here to signal a containing frame that the ATP is being
    * displayed.
    */
  function atpClick()
  {
    var uBroker = this.getUiBroker();

    if (uBroker != null) {
      // tell the server we are done.
  uBroker.showATP();
      rollImage('IMG-atp', BTNIX_ATP, IMAGESPATH + 'czAvail1.gif', IMAGESPATH + 'czAvail0.gif');
    } else {
      // wait for the uibroker.
    }
  }

  /** Function setImage changes the src attribute of an image on the buttons layer.
    * NOTE: Netscape requires the element index to grab the image appropriately.
    */
  function setImage(imgId, elementIndex, newSrc) {
var block = null;
var divId = 'DIV-buttons';

    if (ns4) {
  block = document.layers[divId];
  block.document.images[elementIndex].src = newSrc;
```

```
  } else if (ie4) {
    block = document.all[divId];
  block.all[imgId].src = newSrc;
  }
  }

  /** Function rollImage swaps images to give the buttons a clicked affect.
    * The original image is generally restored with a timeout based on a page level
    * constant.
    *
    * NOTE: Netscape requires the element index to grab the image appropriately.
    */
  function rollImage(id, elementIndex, newSrc, oldSrc)
  {
    setImage(id, elementIndex, newSrc);
    if (oldSrc) {
      resetImageAfterMilliseconds(id, elementIndex, oldSrc, ROLL_RESET_TIMEOUT);
    }
  }

  /** Function resetImageAfterMilliseconds restores images to return buttons
    * to their original images.
    * The original image is gnerally restored with a timeout based on a page level
    * constant.
    */
  function resetImageAfterMilliseconds(id, elementNum, oldSrc, ms)
  {
    var funcStr = "setImage('" + id + "', " + elementNum + ", '" + oldSrc + "')";
    setTimeout (funcStr, ms);
  }

//--> end browser no-script spoof
</script>
</head>
<body bgcolor="#666666">
  <!-- background images -->
    <!-- left tab -->
    <div id="DIV-tableft" nowrap>
      <img src="/OA_MEDIA/czTabL.gif" width="7" height="21">
    </div>

    <!-- center tab -->
  <div id="DIV-tabcenter" nowrap>
      <img src="/OA_MEDIA/czTabC.gif" width="624" height="21">
    </div>
```

```
    <!-- right tab -->
    <div id="DIV-tabright" nowrap>
      <img src="/OA_MEDIA/czTabR.gif" width="7" height="21">
    </div>
  <!-- end background images -->

  <!-- label -->
<div id="DIV-label" nowrap>
    <font face="Arial" pointsize="12pt" color="white">
      <b>Configuration</b>
    </font>
  </div>

  <!-- buttons (anchor/image) layer -->
<div id="DIV-buttons" nowrap>

    <!-- view toggle (Configuration/Summary screens) -->
    <a href="javascript:viewClick();">
      <img name="IMG-view" src="/OA_MEDIA/czSumm0.gif" border="0" WIDTH="73" HEIGHT="15">
    </a>

    <!-- show Availability to Promise (ATP) button -->
    <a href="javascript:atpClick();">
      <img name="IMG-atp" src="/OA_MEDIA/czAvail0.gif" border="0" WIDTH="73" HEIGHT="15">
    </a>

    <!-- add some space between the buttons. -->
     

    <!-- done button -->
    <a href="javascript:doneClick();">
      <img name="IMG-done" src="/OA_MEDIA/czDone0.gif" border="0" WIDTH="37" HEIGHT="15">
    </a>

    <!-- cancel button -->
    <a href="javascript:cancelClick();">
      <img name="IMG-cancel" src="/OA_MEDIA/czCanc0.gif" border="0" WIDTH="50" HEIGHT="15">
    </a>

  </div>

</body>
</html>
```

Glossary of Terms

This glossary for Oracle Configurator is followed by a Glossary of Acronyms

Active Model

The part of Oracle Configurator runtime architecture that processes model structure and rules to create configurations. Interfaces dynamically with the end user Active UI and data.

Active User Interface

The part of Oracle Configurator runtime architecture that provides the graphical views necessary to create configurations interactively. Interfaces with the Active Model and data to give users access to customer requirements gathering, product selection, and customer-centric extensions.

Application Architecture

The software structure of an application at runtime. Architecture affects how an application is used, maintained, extended, and changed.

Architecture

The software structure of a system. Architecture affects how a system is used, maintained, extended, and changed. See also Application Architecture.

Beta

An external release, delivered as an installable application, and subject to system, validation, and acceptance testing. Specially selected and prepared end users may participate in beta testing.

Bill of Material

A list of component items associated with a parent item (assembly) and information about how each item relates to the parent item.

BOM

See Bill of Material.

BOM Item

The nodes imported into the Oracle Configurator Developer Model that correspond to an Oracle BOM.

BOM Model

The imported Model node in the Oracle Configurator Developer that corresponds to Standard Model in an Oracle BOM.

BOM OptionClass

The imported Model node in the Oracle Configurator Developer that corresponds to Option Class in an Oracle BOM.

BOM StandardItem

The imported Model node in the Oracle Configurator Developer that corresponds to Standard Item in an Oracle BOM.

Boolean Expression

An element of a component in the Model that has two options: true or false.

Bug

See Defect.

Build

A specific instance of an application during its construction. A build must have an install early in the project so that application implementers can unit test their latest work in the context of the entire available application.

CIO

See Oracle Configuration Interface Object.

Client

A runtime program using a server to access functionality shared with other clients.

Comparison Rule

An Oracle Configurator Developer rule type to establish a relationship that determines the selection state of a logical item (option, boolean feature, or

list-of-options feature) based on a comparison of two numeric values (numeric features, totals, resources, option counts, or numeric constants). The numeric values being compared can be computed or they can be discrete intervals in a continuous numeric input.

Compatibility Rule

An Oracle Configurator Developer rule type to establish a relationship among features in the Model that specifies the allowable combinations of options. See also, Property-based Compatibility Rule.

Compatibility Table

A type of compatibility relationship where the allowable combination of options are explicitly enumerated.

Component

Represents a configurable element in a product. An element of the Model structure, typically containing features. May correspond to one screen of selections in an Oracle runtime configurator.

Component Set

An element of the Model that contains a number of components of the same type, where each component of the set is independently configured.

Configuration

A specific set of specifications for a product, resulting from selections made in an Oracle runtime configurator.

Configuration Model

The model structure and rules-based content of an Oracle runtime configurator. The configuration model is constructed and maintained using Oracle Configurator Developer, and is interpreted at runtime by the Active Model.

Configuration Rules

The Oracle Configurator Developer logic rules and numeric rules available for defining configurations.

Configurator

The part of an application that provides custom configuration capabilities.

Constraint Rule

An Oracle Configurator Developer rule type to create a logical relationship among features and options. See also Rules.

Contributes to

An Oracle Configurator Developer numeric rule type for accumulating a total value.

Consumes from

An Oracle Configurator Developer numeric rule type for specifying the quantity of a resource used.

CRM

Customer Relationship Management. The aspect of the enterprise that involves contact with customers, from lead generation to support services.

Customer

The person or persons for whom products are configured by end users of the Oracle Configurator or other ERP and CRM applications.

Customer-centric Extensions

See Customer-centric Views.

Customer-centric Views

Optional extensions to core functionality that supplement product selection with rules for pre-selection, validation, and intelligent views. View capabilities include generative geometry, drawings, sketches and schematics, charts, performance analyses, and ROI calculations.

Customer Requirements

The needs of the customer that serve as the basis for determining the configuration of products, systems, and/or services. Also called Needs Assessment.

Data Import

Populating the Oracle Configurator schema with enterprise data from ERP or legacy systems via import tables.

Data Integration Object

Data Integration Object. A server in the runtime application that creates and manages the interface between the client (usually a user interface like the Active User Interface) and the Oracle Configurator schema.

Data Maintenance Environment

The environment in which the Oracle runtime configurator data is maintained.

Data Replication

The activity of downloading and uploading configuration, quote, and order data between the Oracle Configurator schema on the enterprise server and Oracle Configurator Mobile Database on end-user mobile laptop PCs. See also Data Synchronization.

Datasource

A programmatic reference to a database. Referred to by a datasource name, or DSN.

Data Synchronization

A process for matching the data in the Oracle Configurator schema and the data available to client processes such as the Oracle SellingPoint application. See also Data Replication.

Default

The automatic selection of an option based on the pre-selection rules or the selection of another option.

Defaults

An Oracle Configurator Developer logic rule to determine the logic state of features or options in a default relation to other features and options. For instance, if you set A to True by selecting it, B becomes true (selected) if it is available (not false) and can be set to True without contradicting a non-default rule or a previous default setting for B.

Defect

A failure in a product to satisfy the users' requirements. Defects are prioritized as critical, major, or minor, and fixes range from corrections or workarounds to enhancements. Also known as a “bug”.

Defect Tracking

A system of identifying defects for managing additional tests, testing, and approval for release to users.

Deliverable

A work product that is specified for review and delivery.

Demonstration

A presentation of the tested application, showing a particular usage scenario.

Design Chart

An Oracle Configurator Developer rule type for defining advanced Explicit Compatibilities interactively in a chart view.

Design Review

A technical review that focuses on application or system design.

Developer

The tool (Oracle Configurator Developer) used to create configuration models. The person who uses Oracle Configurator Developer to create a configurator. See also Implementer

DIO

See Data Integration Object.

End User

The ultimate user of the Oracle runtime configurator. The types of end users vary by project but may include salespeople or distributors, administrative office staff, marketing personnel, order entry personnel, product engineers, or customers directly accessing the application via web or kiosk.

Enterprise

The systems and resources of a business.

Environment

The arena in which software tools are used, such as operating system, applications, and server processes.

ERP

Enterprise Resource Planning. A software system and process that provides automation for the customer's back-room operations, including order processing.

Excludes

An Oracle Configurator Developer rule type for determining the logic state of features or options in an excluding relation to other features and options. For instance, if you set A to True, B becomes false, since it is not allowed when A is true. If you set A to False, there is no effect on B, meaning it could be true, false, or unknown.

Extended Functionality

A release after delivery of core functionality that extends that core functionality with customer-centric views, more complex proposal generation, discounting, quoting, and expanded integration with ERP, CRM, and third-party software.

Feature

An element of the Model structure. A configurable parameter of a component. Features can either have a value (numeric or boolean) or enumerated options.

Functional Companion

An object associated with a component that supplies methods that can be used to initialize, validate and generate customer-centric views and outputs for the configuration.

Functional Specification

Document describing the functionality of the application based on user requirements.

Incremental Construction

The process of organizing the construction of the application into builds, where each build is designed to meet a specified portion of the overall requirements and is unit tested.

Implementation

The stage in a project between defining the problem by selecting a configuration technology vendor, such as Oracle, and deploying the completed sales configuration application. The Implementation stage includes gathering requirements, defining test cases, designing the application, constructing and testing the application, and delivering it to users.

Implementer

The person who uses Oracle Configurator Developer to build the model structure, rules, and UI customizations that make up an Oracle runtime configurator.

Implies

An Oracle Configurator Developer logic rule type that determines the logic state of features or options in an implied relation to other features and options. For instance, if you set A to True by selecting it, B becomes true, since selecting A implies that B is also selected. If you set A to False by deselecting it, there is no effect on B, meaning it could be true false or unknown based on other relations B participates in. And if you set B to True by selecting it, there is no effect on A, meaning it could be true false or unknown based on other relations A participates in. But if you set B to False by deselecting it, the relation of A implies B is preserved only by having A be false (deselected) as well.

Import Tables

Tables mirroring the Oracle Configurator schema Item Master structure, but without integrity constraints. Import Tables allow batch population of the Oracle Configurator schema Item Master. Import Tables are used in conjunction with extractions from Oracle Applications or legacy data to create, update, or delete records in the Oracle Configurator schema Item Master.

Install

A program that sets up the local machine and installs the application for testing and use.

Integration

The process of combining multiple software components and making them work together.

Integration Testing

Testing the interaction among software programs that have been integrated into an application or system.

Intelligent Views

Configuration output, such as reports, graphs, schematics, and diagrams, that help to illustrate the value proposition of what is being sold.

Item Master

A table in the Oracle Configurator schema containing data used to structure the product. Data in the item master is either entered manually or imported from Oracle Applications or legacy data.

Item Type

A table in the Oracle Configurator schema containing data used to classify the product data in the item master table.

Log File

A file containing errors, warnings and other information output by the running application.

Logic Rules

Logic rules directly or indirectly set the logical state (true, false, or unknown) of features and options in the Model.

There are four (4) primary logic rules: Implies, Requires, Excludes, and Negates. Each of these rules takes a list of features or options as operands. See also Logic, Implies, Requires, Excludes, and Negates.

Maintainability

The characteristic of a product or process to allow straightforward maintenance, alteration, and extension. Maintainability must be built into the product or process from inception.

Maintenance

The effort of keeping a system running once it has been deployed, through bug fixes, procedure changes, infrastructure adjustments, data replication schedules, etc.

Maintenance Guide

A guide for maintaining a specific application or system. The maintenance guide covers all aspects of maintenance described in the generic Maintenance Plan.

Maintenance Plan

A document that outlines what is required for successful maintenance, and who is responsible for all the actions and deliverables of carrying out maintenance on a system.

MDUI

See Model-driven UI.

Mobile Database

See Oracle Configurator Mobile Database.

Model

The entire hierarchical “tree” view of all the data required for configurations, including model structure, variables such as resources and totals, and elements in support of intermediary rules. May consist of BOM Items.

Model-driven UI

The graphical views of the model structure and rules generated by the Active UI to present end users with interactive product selection based on configuration models.

Model Structure

Hierarchical, “tree” view of data in terms of product elements (Models, Products Components, Features, Options, BOM Models, BOM OptionClasses, BOM StandardItems, Resources, and Totals). May include reusable components.

MRP

Manufacturing Resource Planning. A software system and process for monitoring and maintaining the customer's manufacturing systems.

Negates

An Oracle Configurator Developer logic rule type that determines the logic state of features or options in a negating relation to other features and options. For instance, if you set one item in the relationship to True, the other item must be false. And if you set one item to False, the other item must be true.

Node

The place in a Model occupied by a component, feature, option or variable, BOM Model, BOM OptionClass, or BOM StandardItem.

Numeric Rules

Rules that are used to set the global parameters specified in product structuring. See also, Contributes to and Consumes from.

OC

See Oracle Configurator.

Opportunity

The workspace in the Oracle SellingPoint application and Oracle Sales Online in which products, systems, and/or services are configured, quotes and proposals are generated, and orders are submitted.

Option

An element of the Model. A choice for the value of an enumerated feature.

A logical selection made by the end user when configuring a component.

Oracle Configurator

The product family consisting of development tools and runtime applications such as Oracle Configurator schema, Oracle Configurator Developer, Oracle Configurator window, and Oracle SellingPoint application. Also the Oracle runtime configurator variously packaged for use in networked, mobile, or web deployments.

Oracle Configurator Architecture

The application runtime architecture consists of the Active User Interface, the Active Model, and the Oracle Configurator schema or Oracle Configurator Mobile Database. The application development architecture consists of Oracle Configurator Developer and the Oracle Configurator schema, with test instances of an Oracle runtime configurator.

Oracle Configurator Developer

The suite of tools in the Oracle Configurator product family for constructing and maintaining configurators.

Oracle Configuration Interface Object (CIO)

A server in the runtime application that creates and manages the interface between the client (usually a user interface like the Active User Interface) and the underlying representation of model structure and rules in the Active Model.

CIO protocols support creating and navigating the Model, querying and modifying selection states, and saving and restoring configurations.

Oracle Configurator Mobile Database

The runtime version of the standard Oracle Configurator schema that manages data for the configuration model in a mobile deployment. The runtime schema includes customer, product, and pricing data as well as data created during operation of an Oracle Configurator.

Oracle Configurator Schema

The implementation version of the standard Oracle runtime configurator data-warehousing schema that manages data for the configuration model. The implementation schema includes all the data required for the runtime system as well as specific tables used during the construction of the configurator.

Oracle SellingPoint Application

The test application generated by Oracle Configurator Developer. Also a full configuration environment with opportunity management, quotes, and proposals for networked or mobile deployments.

Output

The output generated by a configurator, such as quotes, proposals, bills of material (BOM), and customer-centric views.

PDM

Product Data Management. A software system that manages the version control of product data.

Populator

An entity in the Oracle Configurator Developer that defines how to create a Model from information in the item master.

Pre-selection

The default state in a configurator that defines an initial selection of components, features, and options for configuration.

A process that is implemented to select the initial element(s) of the configuration.

Principal Design Consultant

Member of the project team responsible for architecting the design of the application.

Product

Whatever is subjected to configuration and is the output of the application.

The root element of the Model.

Product Family

A collection of products or product lines, which are organized as a group by a provider or manufacturer.

Project

The workspace in Oracle Configurator Developer in which configurators are constructed

Project Manager

A member of the project team who is responsible for directing the project during implementation.

Project Plan

A document that outlines the logistics of successfully implementing the project, including the schedule.

Property

A named value associated with an object in the Model or the item master. A set of properties may be associated with an item type.

Property-based Compatibility Rule

A kind of compatibility relationship where the allowable combinations of options are specified implicitly by relationships among property values of the options.

Prototype

A construction technique in which a preliminary version of the application, or part of the application, is built to facilitate user feedback, to prove feasibility or examine other implementation issues.

Reference

The use of a reusable component within the Model. Not implemented in Release 11*i* or before.

Regression Test

An automated test that ensures the newest build still meets previously tested requirements and functionality.

Requires

An Oracle Configurator Developer logic rule type that determines the logic state of features or options in a requirement relation to other features and options. For instance, if you set one item in the relationship to True, the other item is required to be true as well. And if you set one item to False, the other item must be false as well.

Resource

Staff or materials available or needed within an enterprise.

A variable in the Model used to maintain the balance of features not consuming more of a specific resource than has been provided by other features.

Reusable Component

A component that is referenced from multiple locations in the Model. Not implemented in Release 11*i* or before.

Reusability

The extent to and ease with which parts of a system can be put to use in other systems.

Rules

Also called business rules or configuration rules. Constraints applied among elements of the product to ensure that defined relationships are preserved during configuration. Elements of the product are components, features, and options. Rules express logic, numeric parameters, implicit compatibility, or explicit compatibility. Rules are used to provide pre-selection and validation capability in an application.

See also Logic Rules and Numeric Rules.

Runtime

The environment and context in which applications are run or used, rather than developed.

Sales Configuration

A part of the sales process to which configuration technology has been applied in order to increase sales effectiveness and decrease order errors. Commonly identifies needs assessment and product configuration.

Server

Centrally located software processes or hardware, shared by clients.

Solution

The deployed system as a response to a problem or problems.

System

The hardware and software components and infrastructure integrated to satisfy functional and performance requirements.

Test Case

A description of inputs, execution instructions, and expected results, which are created for the purpose of determining whether a specific software feature works correctly or a specific requirement has been met.

Total

A variable in the Model used to accumulate a numeric total, such as total price or total weight.

Undetermined

The logic state that is neither true nor false, but unknown at the time a logic rule is executed. This logic state is also referred to as Available, especially when considered from the point of view of the Oracle runtime configurator end user.

Unit Test

Execution of individual routines and modules by the application implementer or by an independent test consultant for the purposes of finding defects.

Update

Moving a production configurator to a new version of configuration model.

Upgrade

Moving the configurator to a new release of Oracle Configurator.

User

The person using the Oracle Configurator Developer tools and methods to build an Oracle runtime configurator. See also end user.

User Interface

The visible part of the application, including menus, dialog boxes, and other on-screen elements. The part of a system where the user interacts with the software.

User Requirements

A description of what the Oracle Configurator or Oracle SellingPoint application is expected to do from the end user's perspective.

User's Guide

Documentation on using the application or configurator to solve the intended problem.

Validation

Tests that ensure that the configured components will meet specific performance or acceptance criteria.

A type of functional companion that is implemented to ensure that the configured components will meet specific performance or acceptance criteria.

Variable

Parts of the Model that are represented by Totals, Resources, or numeric Features.

Verification

Tests that check whether the result agrees with the specification.

Glossary of Acronyms

API

Application Programming Interface

ATP

Available to Promise

BOM

Bill of Material

CIO

Configuration Interface Object

CM

Configuration Management

COM

Component Object Model

CRM

Customer Relationship Management

DBMS

Database Management System

DCOM

Distributed Component Object Modeling

DHTML

Dynamic Hypertext Markup Language

DIO

Data Integration Object

DLL

Dynamically Linked Library

DXF

Drawing Exchange Format (AutoCAD drawings)

ECO

Engineering Change Order

ERM

Enterprise Relationship Management

ERP

Enterprise Resource Planning

ESD

Electronic Software Distribution

ESP

External Service Provider

ESS

Enterprise Selling System

HT

High Tech

HTML

Hypertext Markup Language

IP

Industrial Products

IS
Information Services

ISS
Interactive Selling System

ISV
Independent Software Vendor

LAN
Local Area Network

MAPI
Messaging Application Programming Interface

MC/S
Mobile Client/Server System

MDUI
Model-Driven User Interface

MES
Marketing Encyclopedia System (Catalog)

MIS
Management Information Systems

MRP
Manufacturing Resource Planning

MS
Microsoft

OC
Oracle Configurator

OCX
Object Control File, OLE custom controls

ODBC
Open Database Connectivity

OLE
Object linking and embedding

OMS
Opportunity Management System

OOD
Object-Oriented Design

ORB
Object Request Broker

PDM
Product Data Management

PIA
Project Impact Assessment

POS
Point of Sale

QA
Quality Assurance

RAD
Rapid Application Development

RDBMS
Relational Database Management System

RFQ
Request for Quote

ROI
Return on Investment

SAS
Sales Analysis System

SCM
Supply Chain Management

SCS
Sales Configuration System

SE
Sales Engineer

SFA
Sales Force Automation

SI
System Integrator

SOT
Strategic Options Theory

SQA
Software Quality Assurance

SQL
Structured Query Language

TERM
Technology-Enabled Relationship Management

TES
Technology-Enabled Selling

UI
User Interface

VAR
Value-Added Reseller

VB

Microsoft Visual Basic

WAN

Wide Area Network

WIP

Work In Progress

Index

A

Advanced Pricing, 3–11, 6–1, 6–3
agreement_id, 3–12, 3–14
agreement_type_code, 3–12, 3–14
alt_database_name, 3–15
APPL_TOP, 2–3
application
  tier, 1–2
application logic, 1–2
application server, 1–2
APPLICATION_ID, 3–15
application_id, 3–14
apps_connection_info, 3–11, 3–13, 3–15
apps.zip, 2–5
architecture
  of Oracle Configurator, 1–2
  three-tier, 1–2
ATP
  See Available To Promise
atp_date
  XML element, 4–6
atp_package_name, 3–13, 3–15
atp_timeout, 3–14, 3–15
audience
  of this document, xiii
Available To Promise (ATP)
  integration with, 6–1

B

batch validation message, 5–1
bitmap files, 7–6
BMP files, 7–6
BOM data, 6–3
BOM_EXPLOSIONS, 4–6
Border Frames, 7–2
browsers
  Internet Explorer, 7–4, 7–7
  Netscape, 7–4, 7–7

C

call_atp() procedure, 6–9
callback interface
  for ATP, 6–11
  for pricing, 6–8
  parameters, 6–6, 6–9
callback pricing parameters, 3–12, 3–13
calling_application_id, 3–15
case-sensitivity, 2–3
CIO, 1–6
.class, 2–4
CLASSPATH, 2–4
client
  thin, 1–2
  tier, 1–2
complete_configuration
  XML element, 4–4
COMPONENT_CODE, 4–6
component_code
  XML element, 4–6, 4–8
config_creation_date, 3–10, 3–15
CONFIG_HDR_ID, 3–16
config_header_id, 3–9, 3–16
  XML element, 4–4
CONFIG_ITEM_ID, 4–6
config_messages
  XML element, 4–7

CONFIG_REV_NBR, 3–16
config_rev_nbr, 3–9, 3–16
  XML element, 4–4
config_total_price, 6–6
config.jar, 2–5
configuration window, 1–6
  default user interface, 7–6
CONFIGURATOR_SESSION_KEY, 6–7
configurator_session_key, 3–12, 3–13, 3–16, 6–5, 6–6, 6–9
Configure button, 3–2
Content Frame, 7–2, 7–3
context_org_id, 3–10, 3–16
conventions
  typographical, xiv
customer_id, 3–12, 3–13, 3–16, 6–9
customer_site_id, 3–13, 3–16, 6–9
CZ_ATP_REQUESTS, 6–9
  table description, 6–10
CZ_CF_API, 5–1
CZ_CONFIG_HDRS, 3–16
CZ_CONFIG_ITEMS, 4–6
CZ_PRICING_STRUCTURES, 6–7
  table description, 6–7
CZ_UI_DEFS, 3–20
cz3rdpty.jar, 2–5
czAll.js, 2–6
czCntnt.htm, 2–5, 7–3
czdisp.htm, 2–5, 7–3
cz.dll, 2–5
czFraIE.htm, 2–6, 7–3, 7–4, 7–7
czFraNS.htm, 2–6, 7–3, 7–4, 7–7
czHeader.htm, 2–6, 7–3
czjni.dll, 2–5
czLeft.htm, 2–6, 7–3
czRight.htm, 7–3
czSource.htm, 2–5, 7–3
cztree.htm, 2–5, 7–3

D

database, 1–6
  tier, 1–3
database_id, 3–17
Developer
  editing default user interface, 7–6
DHTML
  representation in UI, 7–1
directories
  HTML, 2–5
  Media, 2–6
  Servlet, 2–4
discounted_price
  XML element, 4–7
.dll, 2–4
document element, 3–4
drivers
  thin required, 3–15
DTD
  for XML elements, 4–2

E

environment variables, 2–4
events, 1–5
exceptions
  data sent to return URL, 3–10
exit
  XML element, 4–4
extensions
  of filenames, 2–4

F

FND_APPLICATION, 3–15
FND_RESPONSIBILITY, 3–19
FND_USER, 3–21
fndnam, 3–5, 3–17
frames
  location property, 3–2

G

Get ATP Dates, 6–9
  ATP interface procedure, 6–9
get_atp_dates_proc, 3–13, 3–17
.gif, 2–4
GIF files, 7–6
gsa, 3–12, 3–17
gwyuid, 3–5, 3–17

H

Header Frame, 7–2, 7–3, 7–7

host application, 1–6
requirements, 3–2
responsibilities, 3–3
.htm, 2–4
.html, 2–4
HTML directory, 2–5
HTML templates, 1–2, 1–5, 1–6, 7–1
Border Frames, 7–2
Content Frame, 7–2, 7–3
customized, 1–6
czFraIE.htm, 7–4, 7–7
czFraNS.htm, 7–4, 7–7
editing tools required, 7–1
files, 3–2
Header Frame, 7–2, 7–7
Inner Frameset, 7–2
Left Frame, 7–2
optional elements, 7–1
Outer Frameset, 7–2, 7–3
Proxy Frame, 7–2, 7–3
required elements, 7–1
Right Frame, 7–2
Source Frame, 7–2, 7–3
structure, 7–1

I

icx_session_ticket, 3–17
image files, 1–7
initialization
definition, 3–3
testing, 3–6
initialization message, 1–2, 1–5, 1–7, 3–2, 3–3, 4–9
for pricing and ATP, 6–3, 6–8, 6–11
syntax, 3–4
validation of parameters, 3–7
initialization parameters
See also XML elements
agreement_id, 3–14
agreement_type_code, 3–14
alt_database_name, 3–15
apps_connection_info, 3–15
arbitrary type, 3–14
atp_package_name, 3–15
atp_timeout, 3–15
calling_application_id, 3–15
config_creation_date, 3–15
config_header_id, 3–16
config_rev_nbr, 3–16
configuration identification type, 3–8
configurator_session_key, 3–16
context_org_id, 3–16
customer_id, 3–16
customer_site_id, 3–16
database_id, 3–17
fndnam, 3–17
get_atp_dates_proc, 3–17
gsa, 3–17
gwyuid, 3–17
icx_session_ticket, 3–17
invoice_to_site_use_id, 3–17
login type, 3–8
model_id, 3–17
model_quantity, 3–18
obtaining list of, 3–14
order_type_id, 3–18
po_number, 3–18
price_list_id, 3–18
price_mult_items_proc, 3–18
price_single_item_proc, 3–18
pricing type, 3–11
pricing_*flex_field*, 3–18
pricing_package_name, 3–18
pwd, 3–19
read_only, 3–19
responsibility_id, 3–19
return URL type, 3–10
return_url, 3–19
save_config_behavior, 3–19
save_usage_behavior, 3–19
ship_to_org_id, 3–20
ship_to_site_use_id, 3–20
template_url, 3–20
terminate_msg_behavior, 3–20
two_task, 3–20
types of, 3–7
ui_def_id, 3–20
ui_type, 3–21
user, 3–21
warehouse_id, 3–21
initialize

XML element, 3–4
Inner Frameset, 7–2, 7–3
installation
servlet, 2–1
Solaris, 2–2
Windows NT, 2–2
Internet Explorer, 7–4, 7–7
INVENTORY_ITEM_ID, 3–17, 4–6
inventory_item_id
XML element, 4–6
invoice_to_site_use_id, 3–12, 3–17
iStore, 6–1
ITEM_KEY, 6–7
item_key, 6–6
ITEM_KEY_TYPE, 6–7
item_name
XML element, 4–8

J

.jar, 2–4
.java, 2–4
JavaScript, 1–5
JavaScript controls
files installed, 2–6
JDBC, 2–2, 3–15
jdbc111.zip, 2–5
.jpg, 2–4
JPG files, 7–6
.js, 2–4

L

LD_LIBRARY_PATH, 2–5
Left Border Frame, 7–3
Left Frame, 7–2
libczjni.so, 2–5
libcz.so, 2–5
LIST_PRICE, 6–8
list_price, 6–6
XML element, 4–6
location property, 3–2
logic, 1–2
engine, 1–6

M

Media directory, 2–6
message
XML element, 4–7
message_text
XML element, 4–8
message_type
XML element, 4–8
MLS
See Multiple Language Support
model_id, 3–10, 3–17
model_quantity, 3–18
MSG_DATA, 6–8, 6–10
msg_data, 6–6
MTL_SYSTEM_ITEMS, 3–16, 3–17, 4–6
Multiple Language Support, 3–20

N

National Language Support, 3–20
Netscape, 7–4, 7–7
NLS
See National Language Support

O

OA_HTML, 2–3, 3–20
OA_MEDIA, 2–3, C–2
OC
See Oracle Configurator
OE_ORDER_LINES_ALL, 3–21
Oracle Applications, 2–1
Oracle Applications 10.7, 6–1
Oracle Applications 11.0, 6–1
Oracle Applications Release 11i, 6–3
Oracle Configurator, xiii
elements of, 1–5
Oracle Configurator Database, 1–3
Oracle Configurator Developer, 1–1
ORACLE_HOME, 2–3
Oracle8 Enterprise Edition, 6–3
Oracle8*i* Enterprise Edition, 1–3, 6–3
order entry
integration, 1–7
order_type_id, 3–12, 3–18
ORGANIZATION_ID, 3–16, 4–6

organization_id
XML element, 4–6
ORIG_SYS_REF, 6–7, 6–10
Outer Frameset, 7–2, 7–3

P

param
XML element, 3–4
parameters, initialization
See initialization parameters
parent_line_id
XML element, 4–6
PATH, 2–4
PL/SQL
application constraints, 5–1
po_number, 3–12, 3–18
positional notation, 6–6, 6–9
POST method, 3–3
Price Multiple Items
description of, 6–6
pricing interface package procedure, 6–5
use of database, 6–7
Price Single Item
description of, 6–5
pricing interface package procedure, 6–5
price_list_id, 3–12, 3–18, 6–4
price_mult_items_proc, 3–12, 3–18
price_single_item_proc, 3–12, 3–18
price_type, 6–5, 6–6
prices_calculated_flag
XML element, 4–5
pricing
integration with, 6–1
interface package procedures, 6–5
list prices, 6–13
selling prices, 6–14
through Advanced Pricing engine, 3–11
through Oracle Applications, 3–11
total price, 6–14
pricing_flex_field, 3–12
pricing_*flex_field*, 3–18
pricing_package_name, 3–12, 3–18
Proxy Frame, 7–2, 7–3
Proxy.class, 7–3
PS_NODE_ID, 6–7
ps_node_id, 6–6
XML element, 4–8
pwd, 3–6, 3–19

Q

QP, 3–11, 6–1, 6–3
QUANTITY, 6–8
quantity, 6–6
XML element, 4–6

R

RapidInstall, 2–2
read_only, 3–19
requested_date, 3–13, 6–9
RESPONSIBILITY_ID, 3–19
responsibility_id, 3–14, 3–19
return URL, 1–2, 1–7, 3–3, 4–3
implementation, 4–9
specification in initialization message, 3–10
submission behavior, 4–3
template code, A–1
return_url, 3–6, 3–10, 3–19
Right Border Frame, 7–3
Right Frame, 7–2

S

save_config_behavior, 3–19
save_usage_behavior, 3–19
scripts, 6–2, 6–4
CZ_ATP_CALLBACK_UTIL_B_80.sql, 6–5
CZ_ATP_CALLBACK_UTIL_B.sql, 6–5
CZ_ATP_CALLBACK_UTIL_S.sql, 6–5
CZ_ATP_UTIL_B.sql, 6–5
CZ_ATP_UTIL_S.sql, 6–5
CZ_LIST_PRICE_B.sql, 6–5
CZ_LIST_PRICE_S.sql, 6–5
CZ_PRC_CALLBACK_UTIL_B_80.sql, 6–5
CZ_PRC_CALLBACK_UTIL_B.sql, 6–5
CZ_PRC_CALLBACK_UTIL_S.sql, 6–5
CZ_PRC_UTIL_B.sql, 6–5
CZ_PRC_UTIL_S.sql, 6–5
GO_IMPORT.sql, 6–4
InstAppsIntegrate.sql, 6–4
InstAppsIntegrateViaLink.sql, 6–4

selection_line_id
XML element, 4–6
SELLING_PRICE, 6–8
selling_price, 6–6
SEQ_NBR, 6–7
server, 1–2
servlet, 1–2, 3–2
installation, 2–1
location, 1–4
URL for, 3–4
Servlet directory, 2–4
SHIP_FROM_ORG_ID, 3–21
ship_to_group_date, 6–9
ship_to_org_id, 3–13, 3–20, 6–9
ship_to_site_use_id, 3–12, 3–20
.so, 2–4
SO_INTERFACE_* tables, 6–4
Solaris
installation on, 2–2
Source and Proxy Frameset, 7–3
Source Frame, 7–2, 7–3
SPSetup.exe, 2–2
syntax
initialization message, 3–4

T

tasks
for using Oracle Configurator, 1–1
template_url, 3–20
terminate
XML element, 4–2
terminate_msg_behavior, 3–20
termination message, 1–2, 1–7, 3–20, 4–1
passed to return URL, 3–10, 4–9
syntax, 4–2
test page, 3–6, 6–11
thin client, 1–2
thin drivers, 3–15
tiers
of Oracle Configurator architecture, 1–2
two_task, 3–5, 3–20

U

UI Server, 1–5, 1–6, 7–1
UI Servlet, 1–5, 1–6, 3–2
installation
See installation
URL for, 3–4
UI_DEF_ID, 3–20
ui_def_id, 3–6, 3–9, 3–20
ui_type, 3–6, 3–21
unsatisfied selections
display of, 7–6
uom
XML element, 4–6
UOM_CODE, 6–8, 6–10
uom_code, 6–6
URL
for UI Servlet, 3–4
US, 3–20
user, 3–6, 3–21
User Interface, 1–1
user interface
customizing, 7–1 to 7–9
generating default, 7–6
modifying default, 7–6
restrictions, 7–6
source of definition, 7–6
USER_ID, 3–21
user_id, 3–14

V

valid_configuration
XML element, 4–4
VALIDATE, 5–1
validation failure objects, 4–7

W

warehouse_id, 3–13, 3–21, 6–9
web application
elements of, 1–5
web server, 1–6
web store, xiii
Windows NT
installation on, 2–2

X

XML, 1–2, 3–4

messages, 1–5
use of quotation marks, 3–5
XML elements
atp_date, 4–6
complete_configuration, 4–4
component_code, 4–6, 4–8
config_header_id, 4–4
config_messages, 4–7
config_rev_nbr, 4–4
discounted_price, 4–7
DTD for, 4–2
exit, 4–4
initialize, 3–4
inventory_item_id, 4–6
item_name, 4–8
list_price, 4–6
message, 4–7
message_text, 4–8
message_type, 4–8
organization_id, 4–6
param, 3–4
parent_line_id, 4–6
prices_calculated_flag, 4–5
ps_node_id, 4–8
quantity, 4–6
selection_line_id, 4–6
terminate, 4–2
structure, 4–2
uom, 4–6
valid_configuration, 4–4
xmlparserv2.zip, 2–5

Z

.zip, 2–4